AFGL-TR-80-0204

LEVEL II

12

A GENERAL SURFACE REPRESENTATION MODULE DESIGNED FOR GEODESY

HANS SÜNKEL

THE OHIO STATE UNIVERSITY
RESEARCH FOUNDATION

JUNE 1980

SCIENTIFIC REPORT NO. 3

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AFB, MASSACHUSETTS 01731

Unclassifed

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (19) (18) AFGL-TR-80-0204 / 2. GOVT ACCESSION NO. AD-A093978	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* (6) A GENERAL SURFACE REPRESENTATION MODULE DESIGNED FOR GEODESY	5. TYPE OF REPORT & PERIOD COVERED Scientific Report No. 3 / 6. PERFORMING ORG. REPORT NUMBER Report No. 292
7. AUTHOR(s) (10) Hans Sünkel (12) 167	8. CONTRACT OR GRANT NUMBER(s) (15) F19628-79-C-0075
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Geodetic Science / The Ohio State University / Columbus, Ohio 43210	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62101F (16) 7600 03AL (17) 03
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Geophysics Laboratory / Hanscom AFB, Massachusetts 01731 / Contract Monitor - Bela Szabo/LW (11)	12. REPORT DATE June 1980 / 13. NUMBER OF PAGES 163 Pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* (14) DGS-292, SCIENTIFIC-3	15. SECURITY CLASS. *(of this report)* Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)* Approved for public release; distribution unlimited	
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*	
18. SUPPLEMENTARY NOTES Tech, Other	
19 KEY WORDS *(Continue on reverse side if necessary and identify by block number)* Spline representation, Graphical representation, Contouring, Prediction, Collocation	
20 ABSTRACT *(Continue on reverse side if necessary and identify by block number)* Technological developments during the last decade have provided geodesists with the capability of obtaining enormous amounts of data on and outside the surface of the earth. In order to understand the general behavior of the data, a graphical representation of the data is useful. This type of representation can be obtained by using GSPP, Geodetic Science Plotting Package, a set of FORTRAN IV subroutines. This report describes the various algorithms of GSPP	

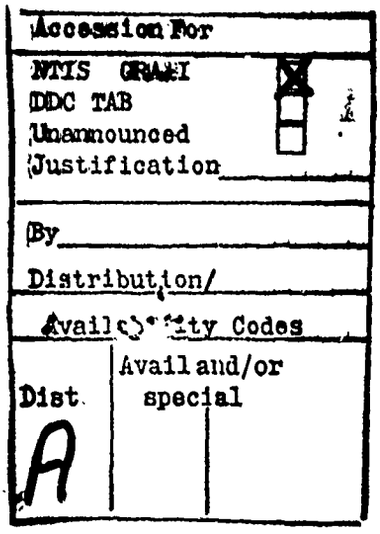
DD FORM 1 JAN 73 1473

Unclassified

used to predict, by a number of different methods, a bicubic spline surface from irregularly distributed data and to produce profiles, contour maps and/or 3-D views of the surface, its first or second derivatives. In addition, complete labelling capabilities are included. Unique features of GSPP are that regions in which contours are to be included/excluded can be defined and that contour maps can be produced according to a user defined projection.

FOREWORD

This report was prepared by Dr. Hans Sünkel, assistant to Dr. Helmut Moritz, Professor, Technical University at Graz and Adjunct Professor, Department of Geodetic Science of The Ohio State University, under Air Force Contract No. F19628-79-C-0075, The Ohio State University Research Foundation, Project No. 711715, Project Supervisor, Urho A. Uotila, Professor, Department of Geodetic Science. The contract covering this research is administered by the Air Force Geophysics Laboratory (AFGL), Hanscom Air Force Base, Massachusetts, with Mr. Bela Szabo, Project Scientist.



CONTENTS

PART A: TECHNICAL BACKGROUND

PART A:

TECHNICAL BACKGROUND

1. INTRODUCTION

Technological developments during the last decades have provided us with sophisticated devices which are able to collect data automatically (satellite altimetry, Doppler methods, satellite to satellite ranging) or semi-automatically (inertial navigation) with an enormous output rate (e.g., almost 10^6 GEOS-3 altimeter data). The tendency is clear: to reduce or eliminate manpower as much as possible in the laborious and expensive data acquisition process. The time seems to be not far away when this first step will have reached its perfection.

In the following steps, however, existing knowledge enters into the data processing stage and with this knowledge enters the human intellect. In order to make data processing faster, computers have been used which were designed in such a way as to permit the solution of many different kinds of problems. A relatively small number of functions is provided by the system. It is usually up to the user to write his own programs for his own purposes. Rigorously tested subroutines are implemented into program libraries. A set of programs might be combined to a large module which would be supposedly capable of handling a whole bunch of problems. Such kinds of modules are, in general, very large (e.g., a satellite orbit prediction module) and not quite transparent to the user; therefore, it is highly desirable that such a module is intelligent by itself. This means that the module should be able to check the input (data, parameters) for consistency, make adjustments if necessary and assign proper default values to undefined parameters. Such a requirement makes it necessary to investigate the way in which decisions are made by intelligent human beings, to abstract this thinking process, and to

translate it into a computer language. It is most interesting and often difficult to split up fully automatic human decision processes into steps or statements and to find the interplay between the visual perception and the processing in the brain. A typical example is the suppressing of drawing contours within certain regions. We shall discuss this problem in more detail in Chapter 2.8.5.

The very complexity of such a module usually makes it hard for the user to understand -- this might be the reason why one speaks frequently about "black boxes". GSPP, the Geodetic Science Plotting Package, is such a black box which is designed for the purpose of graphical representation of data and smooth surfaces. It is a fully automatic link between the stages of data acquisition and interpretation of results. In view of its great complexity and versatility it is very smart and simple to use; this simplicity is mainly due to the control part of GSPP which checks all parameters for consistency, makes necessary corrections, and assigns default values to undefined parameters.

Since one expects GSPP to be fed a large amount of data, all operations have been carefully checked and optimized in terms of CPU-time.

Geodesists are used to dealing with data on and outside the surface of the earth and to preparing contour maps of surfaces like terrain contour maps, gravity anomaly contour maps or geoid contour maps. These two-dimensional representations of surfaces have -- besides providing a general behavior of the surface -- the advantage of allowing the user to also interpolate, to some extent, the information contained in the contours. This makes a contour map superior to a single profile as far as global numerical information is concerned and also superior to a three-dimensional view which provides the user a unique spatial image but lacks numerical information.

Therefore, contour maps are indeed unique and this is the reason why the present report starts with the discussion of contour maps and not with profiles as one would expect.

The user primarily interested in applications may skip the technical Part A which is intended to throw light into the darkness of the black box; he may start with Part B and consult Part A when the need arises.

2. CONTOURING

Before we start with a detailed description of the whole contouring process it should be pointed out that GSPP is not designed for the purpose of representing terrain structures with all its many details as a digital terrain model will do -- it is designed for representing smooth surfaces without artificial structures. Such smooth surfaces are derived from surface data located at the grid points of a regular rectangular grid. In almost all practical applications we are, however, far away from this ideal situation, for three reasons: first, the surface information is sampled at points which are, in general, irregularly distributed (terrain heights, gravity anomalies, geoidal heights); second, the data are often heterogeneous in nature and contain surface information only implicitly (data combinations in physical geodesy); third, the data are usually disturbed by some kind of noise. All these deficiencies make contouring a non-trivial and also non-unique task.

2.1 Data sorting and retrieving

Theoretically, all available data should be used in order to really achieve a prediction with the minimum variance. Practically, this is neither possible because of the enormous amount of geodetic data collected so far, nor is it necessary: a predicted value at a point P depends primarily on the data in the neighborhood; remote data will often contribute very little to the result. Therefore, one accepts the sub-optimal solution and takes only a relatively

small number of data for a single prediction into account. All other data which exceed a certain distance from the prediction point, are not considered.

This would make the calculation of all distances between all data points and all prediction points necessary, a very time consuming task especially when working on the surface of the sphere where "expensive" trigonometric functions are necessary in order to calculate the distance. For this reason, it is absolutely necessary to order the data according to some pre-selected pattern in order to be able to single out the unnecessary part in a simple and fast way. This is also one of the properties of data bases. It is not our intention to establish a sophisticated data base with a very complicated tree structure; all we want to do is to find a simple way of ordering some irregularly distributed data.

For the sake of simplicity, we assume data to be 1-point data. Let (x_i, y_i), $i = 1, \ldots, n$ be the coordinates of a data point P_i in a Cartesian coordinate system. Then an obvious way of arranging the data would be to select a certain "working domain", which is preferably a rectangle, by defining the boundaries of the rectangle (x_L, x_U, y_L, y_U), the lower and upper x- and y-coordinates if the rectangle happens to be parallel to the coordinate lines. This rectangle will then be subdivided into a number of subrectangles. The idea of the data sorting process is to

1. find for each data point the corresponding subrectangle (element),
2. count the number of data within each element, and
3. generate pointer vectors which allow us to find all data points within a specific element.

Note, that all data remain at their original storage locations -- only additional information is produced.

Let us now follow these three steps in detail:

Assume the working domain is subdivided into I * J elements (I in x-, J in y-direction) and enumerate these elements such that an integer (i-1)*J + j is assigned to the element (i,j). Then for each data point there exists an integer which identifies an element if all data are located within the working domain; this vector will be denoted by a; the value of a_k depends only on the position of P_k,

$$a_k = f(P_k) .$$

The length of a is equal to the number of data points. At the same time a counter vector b (length = number of elements) counts the number of data points found in each element. After all data have been located, b contains information about the total number of data within each element.

The problem is to generate pointer vectors such that it is possible to find all data within a prescribed element. For this purpose another vector c has to be generated which has the same length as b and is its partial sum. It provides the information of the starting place in the final pointer vector d such that the first data point in a particular element $\ell = (i-1) * J + j$ has the index

$$m = d[c(\ell)] ,$$

the second data point in the same element ℓ has the index

$$m = d[c(\ell)+1] ,$$

the last data point in element ℓ has the index

$$m = d[c(\ell) + b(\ell) - 1] .$$

If the working domain does not enclose all the data, the vectors b and c have to be longer by 1 element such that the number of data outside the working domain can be stored. In this case $b(I*J+1)$ stores the number of data outside the working domain and $d[c(I*J+1)]$ is the index of the first data point found outside the working domain.

The program responsible for this data organization job is called OAF; it is independent and can be run separately. The time for the organization process depends almost entirely on the number of data and increases linearly with the number of data. In order to give an idea about the efficiency we give one example: to order 100 000 data according to the process described above takes 1.5 sec CPU-time on a AMDAHL 470 V/6-II computer which, in the opinion of the author, is quite fast.

2.2 Prediction

In general, most kinds of data will be irregularly distributed. For many reasons, however, a regular distribution (preferably on a regular rectangular grid) would be welcomed:

a) regularly distributed data admit a simple data management and an easy data retrieving;

b) a regular data distribution (on a rectangular grid) is particularly important if one wants to fit a smooth surface like a bicubic spline function to the data;

c) a regular distribution is necessary for fast post-processing of data using techniques like the Fast Fourier Transform method; but also the standard least-squares methods would gain considerably in terms of computer time by taking symmetries of the covariance matrix into account.

There are a number of possibilities of obtaining predicted (or interpolated) values. We will be discussing only two of them, the inversion-free prediction and least-squares collocation.

2.2.1 Inversion-free prediction. In many cases we face the following situation: there is given a huge number of data, irregularly distributed, of a homogeneous type and almost free of errors; predict a reasonable smooth surface. If one would like to obtain an optimal

solution in the sense of least norm, a least-squares solution is appropriate. Minimizing the mean square prediction error leads to least-squares prediction and least-squares collocation. This method, however, requires an a priori covariance function to be known which is missing in many kinds of problems. Moreover, an inversion of a large matrix or many inversions of small matrices make solutions of such kind of problems too often a very expensive task.

A simple and cheap (however, not the best) alternative is to define the predicted value as a weighted average of all data in the neighborhood of the prediction point. The weights will depend on the distances between data points and the prediction point. One kind of such a prediction is the following:

Let $\{f_i\}$, $i = 1, \ldots, n$ be n homogeneous data; then the predicted value at a point P is given by

$$f_P = \frac{\sum_{i=1}^{n} \frac{f_i}{s_{iP}^q}}{\sum_{i=1}^{n} \frac{1}{s_{iP}^q}} = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i} \qquad (2\text{-}1)$$

with s_{iP} ... distance between data point P_i and prediction point P,

q ... power of prediction.

This kind of prediction has the advantage of exactly reproducing the data and of not involving the calculation of time consuming functions and matrix inversions. This makes it particularly useful for large-scale applications of the type of problems discussed above.

It is quite instructive to compare this kind of prediction with least-squares prediction:

Assume we are given 3 error free geoidal heights and want to predict a geoidal height at a point P. Then the inversion-free prediction gives the result

$$f_P = \frac{1}{c}(\alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3)$$

with α_i, $i = 1, 2, 3$ are weights depending only on the distance between P and all P_i (and not on the distance between P_i and P_j); c is the sum of all α_i. In the case $f = f_1 = f_2 = f_3$ we obtain

$$f_P = \frac{1}{c} \cdot (\alpha_1 + \alpha_2 + \alpha_3) = f \frac{1}{c} = f$$

and the predicted value f_P is independent of the position of the prediction point; consequently, the surface generated will be a horizontal plane. In the case of a general vector $\{ f_i \}$, the surface will not be a plane anymore, however, its trend outside the data region (here a triangle) will be to be a plane with function value equal to the mean value of all data. The reason is that the differences between the weights

$$\delta \alpha = \frac{d}{ds} \left(\frac{1}{s^q} \right) \delta s = -q \frac{\delta s}{s} \alpha$$

tend to zero because of the factor $\frac{\delta s}{s}$.

The least-squares prediction behaves essentially different (C_{ij} ... covariances),

$$f_P = [C_{P1}, C_{P2}, C_{P3}] \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{12} & C_{22} & C_{23} \\ C_{13} & C_{23} & C_{33} \end{bmatrix}^{-1} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}.$$

Under the assumptions made above ($\{f_i\}$ = error-free geoidal heights), the main diagonal elements will be constant and equal to the variance C_0. Assume now for a moment that the correlation length of the covariance function C becomes very small compared with the smallest initial distance between the data; then the covariance matrix

will be highly dominated by the diagonal and its inverse will only slightly deviate from a purely diagonal matrix with diagonal elements being almost equal to the inverse of the variance, $\frac{1}{C_0}$. The covariances C_{P1}, C_{P2}, C_{P3} will be small for large $\overline{PP}_i$; as a consequence the surface will become wavy in between the data points and tend to zero outside the data region. (In the spherical case the surface will be such that its integral over the sphere vanishes.) For larger correlation lengths the predicted surface will become stiffer. In any case, the covariance function controls the surface; the "reproduction" of the surface on the basis of its sampled data consists basically of its "inversion".

Therefore, one main difference between both methods is that in least-squares prediction the surface is generated by actually inverting the surface information; in the inversion-free surface prediction, correlations between the data are completely neglected.

A very interesting light has been thrown recently on the inversion-free prediction (interpolation) in (Sünkel, 1980). It has been shown that (2-1) can be represented in terms of base functions $B_q(x)$,

$$f_P = \sum_{i=1}^{n} f_i B_q (P-P_i) \quad .$$

Each base function $B_q(x)$ can be shown to be identical with the Fourier transform of a polynomial spline of degree q-1. Since the Fourier transform of a polynomial spline approaches a rectangle with increasing degree q, the base functions also approach a rectangle as the power of prediction increases. If $q \to \infty$, the "interpolation" function defined by (2.1) will be a pure step function.

Figure 2.1 gives an impression of what happens if the power of prediction is much too large. The data indicated by "+" are "interpolated" by functions which are very close to step functions (Adigüzel, 1979).

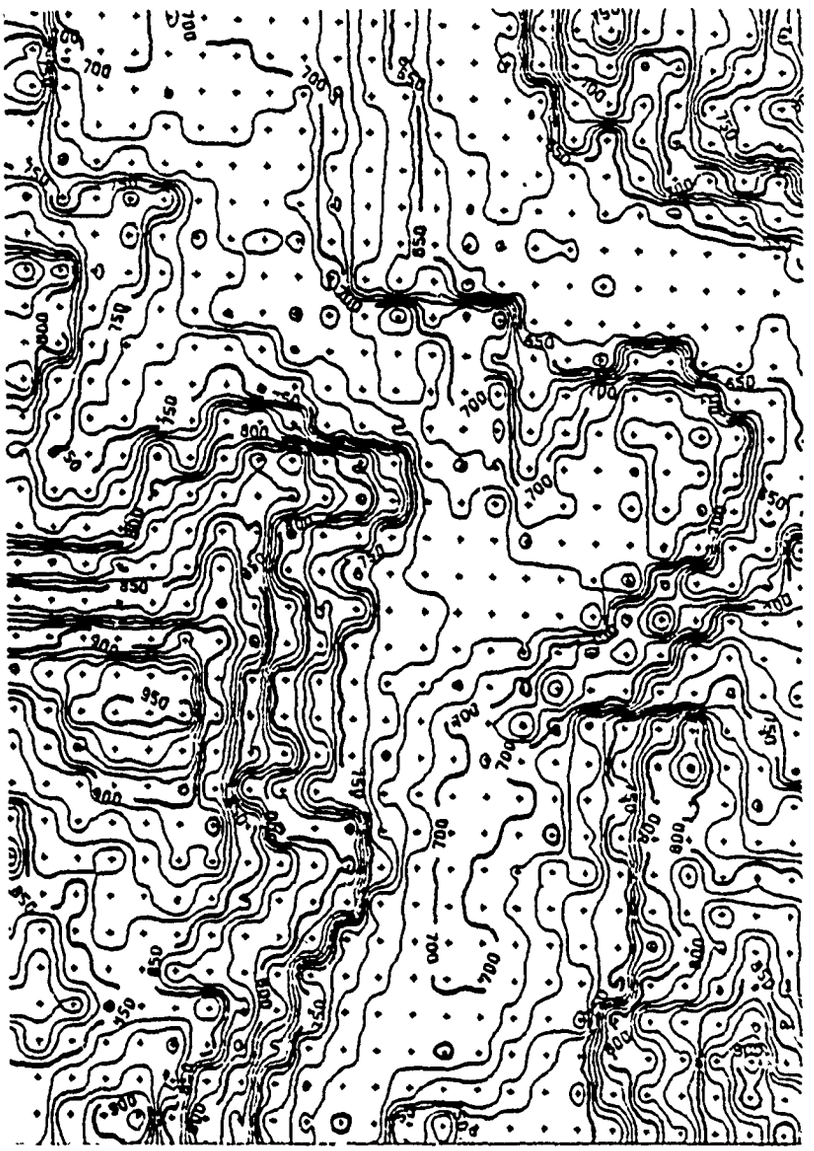


FIG. 2.1 -- Step function like interpolation for very high powers of prediction.

On the other hand, if the power of prediction q is too small, $0 < q \leq 1$, the interpolation function tries to average out all data and, at the same time, it reproduces the data; this causes the function to produce cusps

in the neighborhood of the data points which should obviously be avoided. Since neither steps nor peaks are what one considers as a reasonable interpolation function, one of the often accepted compromises is the choice q = 2 (Schumaker, 1976, Bybee and Bedross, 1978); other users rather prefer a value around 3.5 (Bjerhammar, 1973; Davenport --private communication). GSPP allows the user to make his own choice between the limit values q = 1.5 and q = 4. The method described above goes back to (Shepard, 1964) and (Bjerhammar, 1973).

Another compromise is a splitting of the weight function into three or more parts, each of which shows a different behavior (has different powers q assigned),

$$w(r) = \begin{cases} r^{-1}, & 0 < r \leq \frac{R}{3} \\ \frac{27}{4}\left(\frac{r}{R}-1\right)^2, & \frac{R}{3} < r \leq R \\ 0 & R < r \end{cases}$$

This weight function, proposed by (Shepard, 1964), is continuous and continuously differentiable and vanishes outside r> R, the radius of prediction. Consequently, the choice of R controls the interpolation behavior and, at the same time, defines the prediction circle. (All data within this circle of radius R around the prediction point contribute to the predicted value.) This function is the default function in GSPP if no power of prediction is defined by the user.

2.2.2 Least-squares prediction. The inversion-free prediction as discussed in the foregoing chapter is applicable in a very special case, when data are homogeneous and free of noise. These assumptions are often almost satisfied, however, the general case of heterogeneous noisy data cannot be treated with inversion-free prediction.

As stated in the introduction, the module described here is primarily intended for applications in geodesy, more specifically, for purposes of the determination of the gravity field of the earth. There are many types of data used in physical geodesy; all of them bearing information

about the gravity field, all of them are more or less noisy. The problem is to optimally combine these data in such a way that the gravity field determined on the basis of these data deviates from the true one as little as possible. Least-squares collocation turned out to be particularly useful for the solution of such kind of problems. The goodness of the predicted field depends considerably on the covariance function introduced which should match the average behavior of the gravity field as close as possible (Schwarz and Lachapelle, 1979). For reasons of continuity, the covariance function will be briefly described.

The general form of a homogeneous and isotropic covariance function of the disturbing potential can be expressed by

$$K(P,Q) = \sum_{n=N_0}^{\infty} k_n \left(\frac{R_B^2}{rr'}\right)^{n+1} P_n(\cos\psi_{PQ}) \qquad (2\text{-}2)$$

with P, Q = points outside the sphere $r = R_B$,

r, r' = geocentric radii of P and Q,

ψ_{PQ} = spherical distance between P and Q,

k_n = positive coefficients,

$P_n(\cos\psi)$ = Legendre polynomial of degree n,

N_0 = starting value of the summation ($N_0 \geq 2$),

R_B = radius of the Bjerhammar sphere.

$K(P,Q)$ is symmetric with respect to P and Q and harmonic outside the sphere $r = R_B$.

Geodetic measurements are, in general, nonlinear functionals of the gravity field (+ station position). After linearization they can be expressed as linear functionals of the anomalous gravity field (+ station coordinates).

In order to predict a linear functional of the gravity field at an arbitrary point, it is first of all necessary to establish the covariance

matrix which represents so to say the structural relation between the gravity field and the data. An element of the covariance matrix C_{ij} is the result of an operation which maps the covariance function into the real number line; the mapping consists of applying the linear operations L_i and L_j corresponding to the data i and j on the covariance function K :

$$C_{ij} = L_i L_j K.$$

In the same way are the covariances between the predicted quantity and the data obtained:

$$C_{Pi} = L_P L_i K.$$

Denoting the (linearized) vector of measurements by f as above, one can find a simple linear relation between predicted quantities and data (Moritz, 1978): $f_P = C_P^T C^{-1} f$, or more detailed,

$$f_P = [C_{P1}, C_{P2}, \ldots, C_{Pn}] \begin{bmatrix} C_{11} & C_{12} \ldots & C_{1n} \\ C_{12} & C_{22} \ldots & C_{2n} \\ \vdots & & \vdots \\ C_{1n} & C_{2n} & C_{nn} \end{bmatrix}^{-1} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \quad (2\text{-}3)$$

Taking also noise into account one has to add to C the corresponding error covariance matrix of the data which usually happens to be diagonal. In the general case with incorporated model parameter determination, formula (2-3) changes slightly; however, we shall limit ourselves to the case discussed above. The error covariances between the predicted quantities at the points P and Q can be shown to equal

$$\sigma_{PQ} = C_{PQ} - C_P^T C^{-1} C_Q \quad ;$$

the variance is obtained for Q = P (Heiskanen and Moritz, 1967; p 269 ff.)

The covariances can be derived from a model covariance function which should be simple in order to keep computation time low; on the other hand it should have certain properties (represented by essential parameters) which are determined by the general features of the gravity field. These two requirements exclude each other. Therefore, it is generally a rather time consuming task to find numerical values for all covariances involved, even when using well-designed algorithms (Tscherning, 1976).

Therefore, it has been investigated if one could get rid of this burden by taking advantage of finite elements and approximate the covariance function (Sünkel, 1978). The idea behind this is simple and has been frequently applied in many fields, the network principle: generate a net of fixed points (here grid points in a two-dimensional space) and perform very accurate measurements at these points (here, calculate exact covariances); these fixed points serve as a basis for small scale measurements which can be obtained from simple devices (here, differentiation - interpolation of finite elements representing the covariance function in a certain range). The bicubic spline function turned out to be particularly useful for such a purpose. In (Sünkel, 1979) such a covariance approximation procedure has been described; a FORTRAN IV computer program which has been designed for such purposes, is used by GSPP. This program allows us to obtain covariances of second and lower order derivatives of the disturbing potential at a CPU-time level of some $3 \cdot 10^{-4}$ sec on a IBM 370 system. This advantage, however, is to some extent balanced by disadvantages of one needing: to know the maximum range (in space) in which one is working, to generate a network of covariances and store this network on a (advantageously permanent) file before one can actually call the covariance approximation subroutine; and finally to realize covariances differ slightly from the exact ones derived from the model covariance function; the differences, however, can be kept arbitrarily small. This is the price we pay for a 90 %

reduction in the computation time for covariances.

It should be clear that such sophisticated procedures require decision making before actually using the module GSPP. This means that the covariance approximation procedure should be used only for large scale applications -- for such a purpose it was designed. In such cases it really pays back by saving enormous amounts of computer time. For medium and small scale applications one can use the exact covariances derived from some model.

Detailed information about the covariance approximation algorithm can be found in (Sünkel, 1979); the principle can be briefly described in the following:

An important property of the covariance function which makes a two-dimensional approximation for theoretically all points outside the Bjerhammar sphere possible, is its dependence on essentially two variables, the spherical distance ψ and the product rr'. Since $\cos\psi$ can vary only between -1 and +1 and R_B^2 / rr' has a minimum value of 0 for $r \to \infty$ and a maximum of 1 for $r = r' = R_B$, the covariance function's domain of definition is the rectangle

$$[-1 \leqq t \leqq 1 \, , \; 0 < s < 1]$$

with

$$t = \cos\psi \, , \quad s = \frac{R_B^2}{rr'} \, .$$

Since practically all geodetic operations are performed on or close to the surface of the earth, the domain of definition is reduced considerably for all practical applications, e.g., working within a spherical distance range of $0 \leq \psi \leq 10^{\circ}$ and within an altitude range from 0 to 300 km, the domain reduces to

$$[\, 0.985 \leq t \leq 1 \, , \; 0.999 \geq s \geq 0.912] \, .$$

Once the user has made his decision about the ranges in which he is going to work, a rectangular grid in s and t can be arranged. A

specific program has been designed for the calculation of different kinds of covariances at all grid points. This network of covariances has to be calculated. The calculation itself is a three stage process: in the first part the grid element corresponding to the particular data location has to be found; in the second stage the necessary interpolations and differentiations of the covariance network are performed; and in the last part the program calculates the partial derivatives of cos ψ with respect to the spherical coordinates φ, λ, φ', λ' at the particular points P and Q. So far as covariance approximation is concerned.

The prediction program itself has control over the prediction region, a rectangle surrounding the prediction point. This region can be defined by the user in terms of a radius of prediction. As long as the program finds a minimum number of data for a single prediction, it is fine; if this is not the case the prediction region will be enlarged until a minimum number of data have been found. Three different kinds of prediction are possible, the inversion-free prediction for homogeneous and error-free data, least-squares prediction with accurate covariances derived from a model covariance function, and least-squares prediction with approximated covariances. In the case of least-squares prediction, the system returns, apart from the predicted grid, also the root mean square prediction errors. The calculation of these errors can be suppressed. The predicted grid represents the surface insofar as the bicubic spline interpolation function based on this grid is interpreted as the surface in consideration.

2.3 Least-squares regression

A surface predicted by one of the methods discussed above is capable of representing even small and local details of the surface if the data contain such an information. Such detailed representations can only be described by a large number of parameters; in case of least squares prediction the number of parameters (coefficients) is equal to the number of data. For many reasons, however, one is often not

only interested in the local details but also in the global features of the field represented by the data. Such global features can be described by a relatively small number of parameters. The problem consists of the selection of the model, which is usually a polynomial of some degree, in establishing the relation between the data and the model parameters and in the solution of the linear system of a size equal to the number of parameters. The most obvious solution is a least-squares solution. In this context we speak about least-squares polynomial approximations or simply about least-squares regression. Typical examples of practical applications are trend eliminations in all natural sciences (e.g., determination of spherical harnomic coefficients up to and including degree and order N based on data sensitive to the earth's gravity field).

The basic idea comes from an old interpolation theorem, the century-old Weierstrass approximation theorem and the least-squares principle. Very loosly speaking the first theorem says that all n coefficients of a polynomial model can uniquely be determined from n independent data linearly related to the model. Weierstrass' theorem asserts that a continuous function can be uniformly approximated by a polynomial on a closed interval. The least-squares principle guarantees the uniqueness and existence of a shortest distance between a point (vector of data) and the hyperplane spanned by the linearly independent base functions (polynomials 1, x, y, x^2, xy, y^2, ..., y^k).* The parameters $\{a_i\}$ can be immediately found from the solution of the normal equations:

$$< F - \Phi a , \Phi > = 0 \qquad (2.3\text{-}1)$$

with $F = \{f_i\} = \{L_i f\}$ ... data (L...linear functional)

$a = \{a_i\}$ ... polynomial coefficients

$\Phi = \{\Phi_{ij}\} = \{L_i \varphi_j\}$ ... design matrix

(φ_j ... base functions).

* (Davis, 1975, pp. 24 ff., 107 ff., 158 ff.)

With an a priori error covariance matrix P^{-1} of the data, the well known solution for the parameters is

$$a = (\Phi^T P \Phi)^{-1} \Phi^T P F \quad ; \qquad (2.3\text{-}2)$$

the individual data reproduction errors are given by

$$\Delta F = F - \Phi a. \qquad (2.3\text{-}3)$$

If the number of parameters is equal to the number of data, ΔF vanishes identically (simple interpolation).

In order to avoid any misinterpretation, it should be pointed out that least-squares polynomial approximations do by no means replace or compete with prediction solutions based on the least-squares collocation principle or any other prediction method; they supplement these solutions insofar as they provide a trend information. Such trend calculations may be quite useful for a number of problems; however, the user should be warned not to work with a high degree trend polynomial and to make sophisticated interpretations on the basis of the results: polynomials show the tendency to oscillate between data and do not hesitate to show completely abnormal features in data free regions. (see as an example Fig. 2.2).

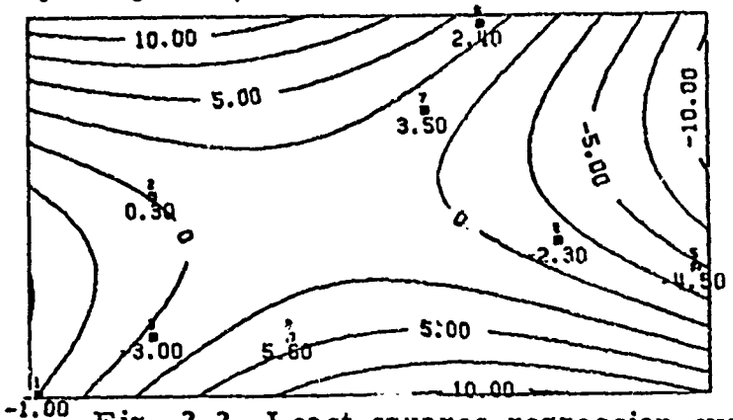


Fig. 2.2 Least-squares regression example

2.4 Smooth surface representation

To predict values at points where no measurement has been performed, is usually very expensive in terms of computer time. The

reasons are multifold: it is necessary to compute distances between each calculation point and theoretically all data points (this deficiency is shared by all prediction methods); the use of least-squares collocation usually involves the calculation of trigonometric and/or logarithmic functions, and the multiplication of large vectors with large matrices for each single prediction. (We don't consider at this point the inversion of the covariance matrix because this is independent on the number of predictions.) Therefore, it is absolutely necessary to find a representation of the surface which is

a) based on the data and/or the predicted values,

b) smooth,

c) local,

d) simple.

Requirement (a) is evident; a sufficient degree of smoothness is desired in order to admit surface differentiations (slope maps, etc.), the interpolating function should be sensitive to a point disturbance only in its neighborhood which is referred to as a local behavior; last, but by no means least, should the interpolating function be simple in order to make the interpolation/differentiation process fast.

Naturally, there does not exist a function which fulfills all these requirements fully. Single polynomials are not local, linear interpolating elements are not smooth. An optimal compromise is possibly a bicubic spline function which is sufficiently smooth (continuous second order derivatives), is strictly local, and still a relatively simple interpolating element. A disadvantage, however, is that bicubic spline functions are based on a regular rectangular grid; smooth surfaces based on irregularly distributed data are possible; the computational effort, however, is huge. On the other hand, a regular rectangular grid of data is anyway the most natural way of storing two-dimensional information, and therefore, this restriction looses much of its power. In the following we give for the sake of completeness a short description of spline interpolation in one and two dimensions which is a

short outline of (Sünkel, 1980) and the procedure GSPP uses to smoothly represent a surface based on gridded data.

2.4.1 One-dimensional cubic spline.

The one-dimensional cubic spline is a basic tool not only for representing a smooth curve, but also for surface fitting. The cubic spline is just one of an infinite number of splines; its pleasant properties make it unique among all splines. It is a function which is twice continuously differentiable and therefore very smooth. Basic for the cubic spline is the cubic basis-spline, or simply B-spline, defined on a grid with constant grid spacing equal to 1. (In such a case one usually speaks of cardinal splines because of its definition on the sequence of cardinal numbers.) Such a cardinal B-spline is a piecewise cubic polynomial with bounded support; it is twice continuously differentiable on the whole real line $(-\infty, \infty)$. Centered at the zero point, it can be expressed by

$$B(x) = \frac{1}{6} \sum_{k=0}^{4} (-1)^k \binom{4}{k} (x + 2 - k)_+^3 \qquad (2.4\text{-}1)$$

with

$$x_+ = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

(I.J. Schoenberg, 1973, p. 11). Its support is the open interval $(-2, 2)$. Explicitly written, $B(x)$ satisfies the equations

$$B(x) = \frac{1}{6} \begin{cases} 3|x|^3 - 6x^2 + 4 & \text{for } 0 \le |x| \le 1 \\ -|x|^3 + 6x^2 - 12|x| + 8 & \text{for } 1 \le |x| \le 2 \\ 0 & 2 \le |x| \end{cases}$$

It can be seen immediately that $B(x)$ is symmetric: $B(x) = B(-x)$.

Its function values at the knots are

$$B(\pm 2) = 0, \quad B(\pm 1) = \frac{1}{6}, \quad B(0) = \frac{4}{6} \qquad (2.4\text{-}2)$$

Figure 2.3 shows the cubic cardinal B-spline together with its derivatives up to and including order 3 which is a step function.

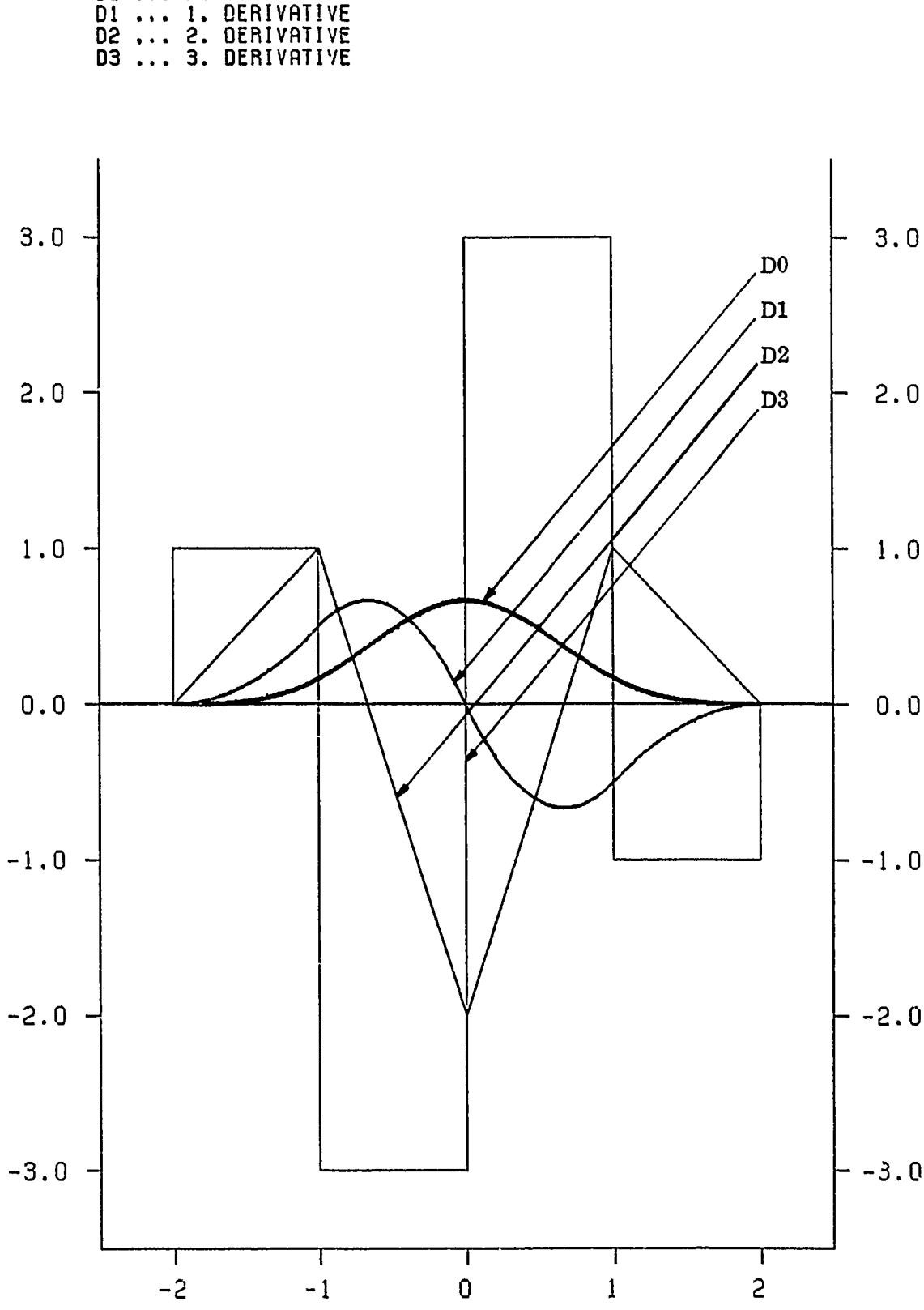


Fig. 2.3 Cubic cardinal B-spline with derivatives

Given function values f_k and second derivatives f_k'' at all integers on the real line, it can easily be shown that

$$F(x) = \sum_{-\infty}^{\infty} c_k B(x-k) \qquad (2.4-3)$$

reproduces the data and is a unique interpolating function (Sünkel, 1980). The coefficients $\{c_k\}$ can, in fact, be expressed by $c_k = f_k - \frac{1}{6} f_k''$. In almost all practical applications, however, the second derivatives are not known. Therefore, the question arises whether it is possible to find an interpolating cubic spline which is only defined on the function values $\{f_k\}$ at the grid points. Or, formulated differently, one would like to have functions $S(x)$ such, that

$$F(x) = \sum_{-\infty}^{\infty} f_k S(x-k) \qquad (2.4-4)$$

is a cubic interpolating spline. Such a function does exist (I.J. Schoenberg, 1973); it can be expressed as a discrete convolution of the form

$$S(x) = \sum_{-\infty}^{\infty} \sigma_j B(x-j) \qquad (2.4-5a)$$

the coefficients $\{\sigma_j\}$ can be determined from the condition that

$$S(k) = \delta_k = \begin{cases} 1 \text{ for } k = 0 \\ 0 \text{ for } k \neq 0 \end{cases} . \qquad (2.4-5b)$$

Since the cubic B-spline has non-vanishing function values at only 3 knots $(-1, 0, 1)$, it follows with (2.4-5a, b) that the infinite sum reduces to a sum over only 3 B-splines for $S(x) = S(k)$:

$$S(k) = \sigma_{k-1} B(1) + \sigma_k B(0) + \sigma_{k+1} B(-1), \quad \forall\, k ,$$

or explicitly ,

$$
\begin{aligned}
&\vdots \\
S(-2) &= \sigma_{-3}B(1) + \sigma_{-2}B(0) + \sigma_{-1}B(-1) = 0 \\
S(-1) &= \sigma_{-2}B(1) + \sigma_{-1}B(0) + \sigma_{0}B(-1) = 0 \\
S(0) &= \sigma_{-1}B(1) + \sigma_{0}B(0) + \sigma_{1}B(-1) = 1 \qquad (2.4\text{-}6) \\
S(1) &= \sigma_{0}B(1) + \sigma_{1}B(0) + \sigma_{2}B(-1) = 0 \\
S(2) &= \sigma_{1}B(1) + \sigma_{2}B(0) + \sigma_{3}B(-1) = 0 . \\
&\vdots
\end{aligned}
$$

Therefore, in order to find the infinite vector of coefficients $\{\sigma_j\}$, $j = -\infty, \ldots, \infty$, we have to solve the infinite system of equations (2.4-6) which, with B(k) from (2.4-2), is given by

$$
\begin{bmatrix}
\ddots & \ddots & \ddots & \ddots & \ddots & & \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
\ldots 0 & 1 & 4 & 1 & 0 \ldots & & \\
& \ldots 0 & 1 & 4 & 1 & 0 \ldots & \\
& & \ldots 0 & 1 & 4 & 1 & 0 \ldots \\
& & & \ldots 0 & 1 & 4 & 1 & 0 \ldots \\
& & & & \ddots & \ddots & \ddots & \ddots & \ddots
\end{bmatrix}
\begin{bmatrix} \vdots \\ \sigma_{-2} \\ \sigma_{-1} \\ \sigma_{0} \\ \sigma_{1} \\ \sigma_{2} \\ \vdots \end{bmatrix}
=
\begin{bmatrix} \vdots \\ 0 \\ 0 \\ 6 \\ 0 \\ 0 \\ \vdots \end{bmatrix} . \qquad (2.4\text{-}6)'
$$

The transformation matrix is of infinite dimension, is symmetric, and because of its "row-shift" structure of Tòeplitz form, it is circulant. Such kind of matrices are well known to have inverses of the same type (R.M. Gray, 1971); with the help of Fourier techniques it is fairly easy to find the solution vector σ. In (H. Sünkel, 1980) it is shown in detail how the actual solution can be obtained; the result is

$$\sigma_j = \sqrt{3}\,(-2 + \sqrt{3})^{|j|} \qquad (2.4\text{-}7)$$

Consequently, the cubic cardinal spline with function values equal to

zero apart from the zero point, where it assumes the value 1, is given by

$$S(x) = \sqrt{3} \sum_{-\infty}^{\infty} (-2 + \sqrt{3})^{|j|} B(x-j). \qquad (2.4\text{-}8)$$

Its behavior can be judged from the graph shown in Figure 2.4:

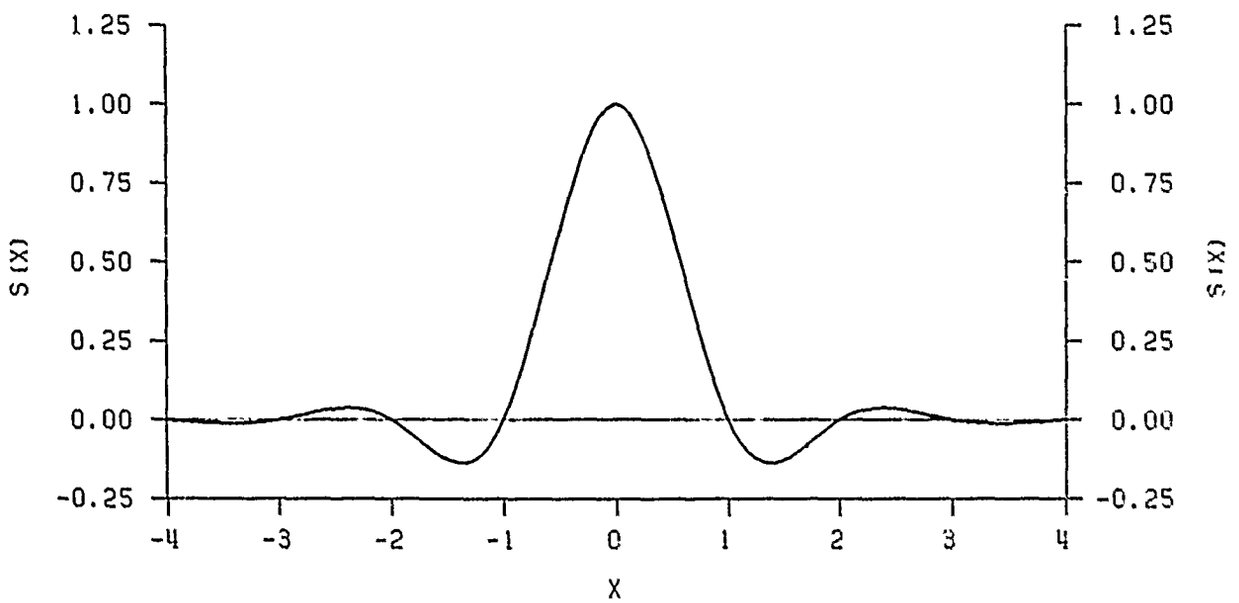


Fig. 2.4 Fundamental Cardinal Cubic Spline

This spline which is known as "fundamental cardinal cubic spline" has unbounded support, is twice continuously differentiable, consists of

cubic polynomials within each interval, is symmetric and interpolates the infinite data vector [..., 0, 0, 1, 0, 0, ...]. Each data vector different from the above one can uniquely be spline-interpolated by a linear combination of fundamental splines with the function values at the grid points as coefficients,

$$F(x) = \sum_{-\infty}^{+\infty} f_k \, S(x-k) \quad .$$

In the discussion above we have limited ourselves to cardinal cubic spline interpolation on the whole real line. This is for many reasons a restriction:

a) in all practical problems data are given on a limited part of the real line;

b) often the data are not regularly distributed with grid distance = 1;

c) cubic spline interpolation is just one kind of spline interpolation; why not use another one, say quadratic or quintic?

There is a good reason to discuss very isolated cases: because they show up the very behaviour of the general solution. This is why cardinal spline interpolation has been discussed and not spline interpolation on an irregular limited grid of data. In such cases the formulas are no longer as simple (the matrix in (2.4-6)' is not as regular, but still tri-diagonal) -- the features, however, remain the same. As far as point (c) is concerned, there is a simple answer: cubic spline interpolation has been chosen not only as a compromise between linear interpolation and spline interpolation of highest smoothness (which is a $\sin x/x$ - interpolation as shown in (H. Sünkel, 1980)); it has also been chosen for serious practical reasons: the cubic spline still retaines a high degree of simplicity, while its attractive features (smoothness, localness), usually adherent to more sophisticated interpola-

tion functions, remain. Two of its properties are worth being at least mentioned: the cubic spline minimizes the overall squared second derivatives (which, in the case of small first derivatives, approximately corresponds to a minimization of the overall curvature and, therefore, the elastic energy),

$$\int [F''(x)]^2 \, dx = \min. \qquad (2.4\text{-}9)$$

among all possible interpolating functions. This property is called the minimum norm property. The second one, called best approximation property, guarantees that the interpolating spline has smaller distance from a given function (sampled at the data points) than any other non-interpolating spline; distance, in this context is defined via the pseudo norm (2.4-9).

As far as approximation properties are concerned, the following error estimates can be shown to hold (Ahlberg et al., 1967):

$$|F^{(\alpha)}(x)-f^{(\alpha)}(x)| \leq h^{3-\alpha} \cdot \int |D^4 f(x)| \, dx, \quad \alpha = 0, 1 \qquad (2.4\text{-}10)$$

with h denoting the grid distance and f(x) the function to be approximated by the cubic spline F(x). Similar error bounds hold for second and third derivatives.

2.4.2 Two-dimensional cubic spline.

Analogous to the one-dimensional case one can define a function of the independent variables x and y, interpolating all data on an infinite regular rectangular grid with constant grid distance equal to 1 (cardinal grid) such that the interpolating function is twice continuously differentiable with respect to both independent variables x and y:

$$F(x, y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} c_{kl} \, B(x-k) \, B(y-l). \qquad (2.4\text{-}11)$$

As in the one-dimensional case, the coefficients $\{c_{kl}\}$ are not simply the function values at the grid points, but linear combinations of function values with derivatives. The grid information is hardly ever available and, therefore, one intends to find base functions such that the function values at the grid points are identical with the coefficients $\{c_{kl}\}$ in (2.4-11). This is in fact possible; the unique fundamental bicubic cardinal spline function is just a product of the one-dimensional splines discussed in Chapter 2.4.1:

$$S(x,y) = \sum_{-\infty}^{+\infty} \sum_{-\infty}^{+\infty} \sigma_i \sigma_j B(x-i) B(x-j). \qquad (2.4\text{-}12)$$

This spline which is known as "fundamental cardinal bicubic spline" has unbounded support, is twice continuously differentiable with respect to x and y, consists of bicubic polynomials within each cell of the grid, and interpolates the infinite data array $\delta_k \delta_l$:

$$\begin{array}{ccc} \vdots & \vdots & \vdots \\ \ldots 0 & 0 & 0 \ldots \\ \ldots 0 & 1 & 0 \ldots \\ \ldots 0 & 0 & 0 \ldots \\ \vdots & \vdots & \vdots \end{array}$$

Its behavior is similar to that of the one-dimensional spline . Figure 2.5 shows its main features.

Each data array which is different from the above one can uniquely be spline-interpolated by a linear combination of such fundamental splines with the function values at the grid points as coefficients,

$$F(x,y) = \sum_{-\infty}^{+\infty} \sum_{-\infty}^{+\infty} f_{kl} S(x-k) S(y-l). \qquad (2.4\text{-}13)$$

As before we have limited ourselves to the presentation of cardinal cubic splines defined on the whole two-dimensional plane. This ideal case will never be met in practical applications; a bounded support

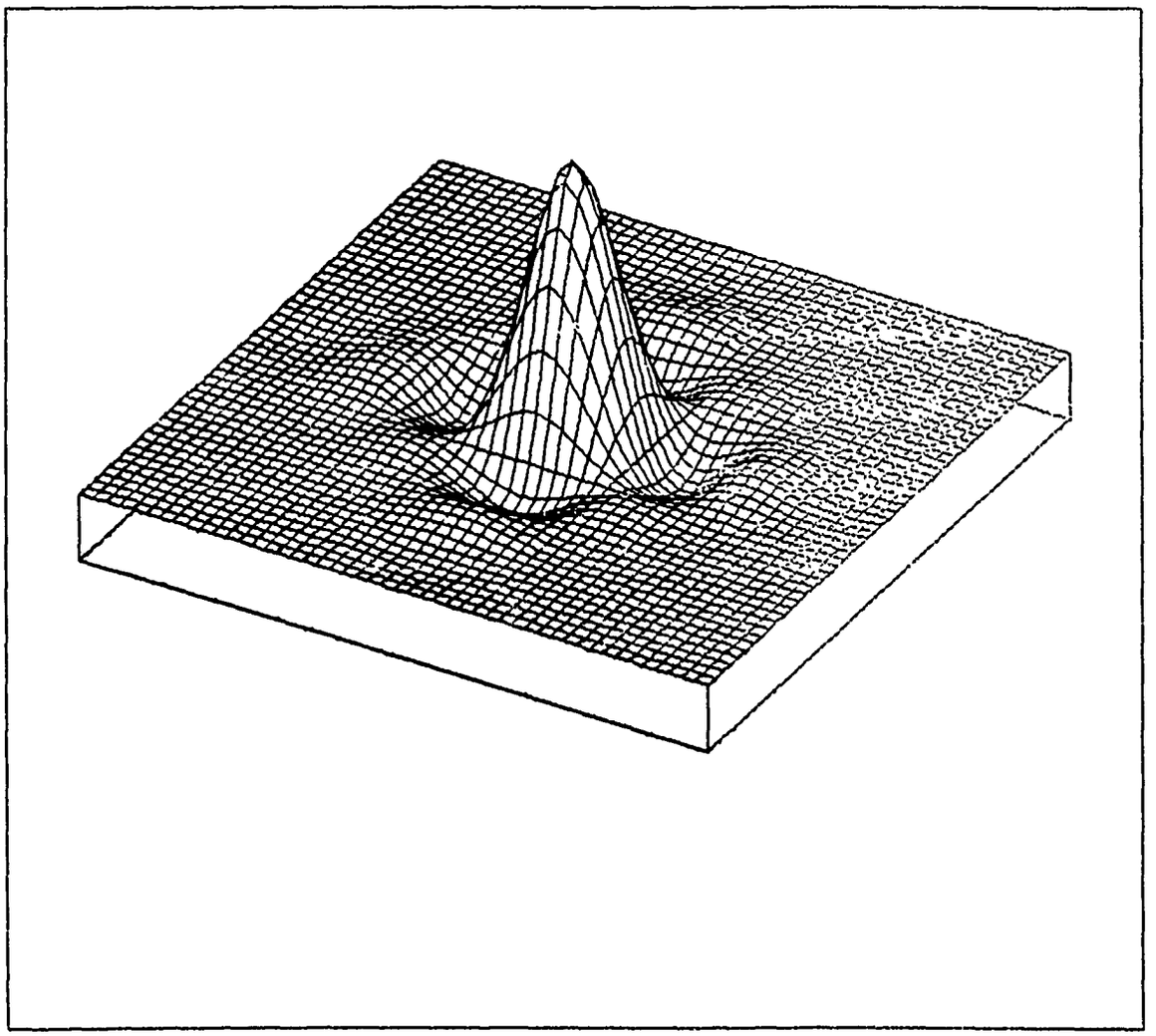


Fig. 2.5 Fundamental bicubic cardinal spline

causes the formulas to become slightly more complicated, but the main features are retained. A non-uniform grid is also possible, however, it has to be generated by lines parallel to the coordinate lines. GSPP does not deal with such a case.

As far as minimum properties are concerned, there is also an analogy with the one-dimensional spline) the bicubic spline minimizes the following integral:

$$\iint \left[\frac{\partial^4 F(x,y)}{\partial x^2 \, \partial y^2} \right]^2 dxdy = \min. \qquad (2.4\text{-}14)$$

among all possible interpolating functions (minimum norm property) and the interpolating spline has least distance (where "distance" is defined via the norm (2.4-14)) from a given function, sampled at the grid points, than any other non-interpolating spline (Ahlberg et. al., 1967).

As far as approximation properties are concerned, it can be shown that the approximation error depends on the grid spacing in the following way:

$$\left| \frac{\partial^\gamma F(x,y)}{\partial x^\alpha \, \partial x^\beta} - \frac{\partial^\gamma f(x,y)}{\partial x^\alpha \, \partial x^\beta} \right| = O(h_x^{3-\alpha} + h_y^{3-\alpha}) ,$$

$$\gamma = \alpha + \beta \leq 6 \quad , \quad \alpha \leq 3, \beta \leq 3$$

and h_x , h_y grid distances in x and y-direction.

2.5 The frequency content of a bicubic spline surface

A bicubic spline surface is based on data (function values) distributed on a regular rectangular grid. Formulated differently, the bicubic interpolating spline is a smooth function interpolating all samples of the original function.

It is well known from the sampling theorem (see e.g. E.O. Brigham, 1974, p. 83 ff) that the original function can be exactly reconstructed from the samples, only if the original function is band-limited with highest frequency f_{max}, and the sampling interval h is smaller than or equal to $1/2\ f_{max}$; the frequency $1/h = 2f_{max}$ is called the Nyquist sampling rate.

In general, the sampled functions are not band-limited and no interpolation function is able to exactly reproduce the original; this fact is called aliasing.

To know which frequencies can be represented by a cubic interpolating spline is of interest by itself; moreover, since operations like differentiations are performed not on the data but on the interpolated values, it is important to know how the interpolated values respond under such operations; this can be seen best in the frequency domain.

In order to study the frequency behavior of bicubic splines, we remember that, according to equation (2.4-13), the two-dimensional spline base function is just a product of one-dimensional splines. Therefore, we investigate now the frequency behavior of the one-dimensional spline.

2.5.1 Spectrum of the cubic spline. We define the spectrum of a function f(x) as its Fourier transform

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega} \, dx \quad , \qquad i = \sqrt{-1} \tag{2.5-1a}$$

with its inverse

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} d\omega \quad . \tag{2.5-1b}$$

We are interested in the case of f(x) to be a cubic spline. There are at least two ways of approach: a delicate and mathematically rather involved one which starts with the spectrum of the B-spline, takes advantage of the properties of Euler-Frobenius polynomials, and derives the transform of fundamental cardinal cubic splines; this approach can be found in detail in (H. Sünkel, 1980). Here a much simpler and straight forward derivation will be given which is based on formulas derived in (H. Sünkel, 1977a). In the following it will be assumed that data are given at all integers from $-\infty$ to $+\infty$ and that a cubic spline should be fitted to these data. Furthermore, the data should be such that the integral (2.5 -1a) exists. Since we know from Chapter 2.4 that the cubic spline consists of cubic polynomials within each interval (between two consecutive data), we can split up the integral (2.5-1a) in a countable

infinite number of integrals, each one being extended over one interval of length 1. Consequently, the contribution of the cubic polynomial defined between the integers m and m+1 is given by

$$F^{(m)}(\omega) = \int_m^{m+1} f^{(m)}(x)\, e^{-i\omega x} dx$$

$$= \int_m^{m+1} f^{(m)}(x)\, \cos\omega x - i \int_m^{m+1} f^{(m)}(x)\, \sin \omega x \qquad (2.5\text{-}2)$$

with the cubic polynomial $f^{(m)}(x)$. Taking (2.4-...) into account, equation (2.5-2) can be split up further into

$$F^{(m)}(\omega) = \sum_{k=0}^{3} a_k^{(m)} \left[C_k^{(m)}(\omega) - iS_k^{(m)}(\omega)\right] \qquad (2.5\text{-}3a)$$

with $\{a_k^{(m)}\}$ being polynomial coefficients, and $C_k^{(m)}(\omega)$ and $S_k^{(m)}(\omega)$ the real and imaginary part of the k^{th} degree polynomial contribution for the interval [m, m+1]:

$$\begin{Bmatrix} C_k^{(m)} \\ S_k^{(m)} \end{Bmatrix} = \int_m^{m+1} (x-m)^k \begin{Bmatrix} \cos \omega x \\ \sin \omega x \end{Bmatrix} dx; \qquad (2.5\text{-}3b)$$

the result of these simple integrations can be found in (H. Sünkel, 1977a, p. 37 ff.):

$$C_0^{(m)}(\omega) - iS_0^{(m)}(\omega) = -\frac{1}{i\omega}\left[e^{-i\omega(m+1)} - e^{-i\omega m}\right],$$

$$C_k^{(m)}(\omega) - iS_k^{(m)}(\omega) = \frac{k}{i\omega}\left[C_{k-1}^{(m)}(\omega) - iS_{k-1}^{(m)}(\omega)\right] - \frac{1}{i\omega} e^{-i\omega(m+1)},$$

$$k = 1, 2, \ldots \qquad (2.5\text{-}4)$$

After some manipulations it is possible to express the contribution of a cubic polynomial, between the interval [m, m+1], to the spectrum $F(\omega)$ in the form

$$F^{(m)}(\omega) = \sum_{k=0}^{3} a_k^{(m)} e^{-i\omega m} \frac{k!}{(i\omega)^{k+1}} \left[1 - e^{-i\omega} \sum_{r=0}^{k} \frac{(i\omega)^{k-r}}{(k-r)!}\right]. \qquad (2.5\text{-}5)$$

The four polynomial coefficients $\{a_k^{(m)}\}$, $k=0,\ldots,3$, however, depend linearly on the function values f_m, f_{m+1} and the derivatives f'_m, f'_{m+1}. The goal is to figure out all coefficients of f_m, f_{m+1}, f'_m, and f'_{m+1} and finally to express f'_m and f'_{m+1} by the infinite vector $\{f_m\}$, $m = -\infty, \ldots, \infty$.

Evaluating (2.5-5) explicitly, one finds the "contribution" of all f_m to be

$$\sum_{-\infty}^{\infty} f_m e^{-i\omega m} \left\{\frac{12}{\omega^4}(1-e^{-i\omega}) - \frac{i}{\omega}\left[1+\frac{6}{\omega^2}(1+e^{-i\omega})\right]\right\} \quad ; \qquad (2.5\text{-}6a)$$

since $\sum_{-\infty}^{\infty} f_{m+1} e^{-i\omega m} = e^{i\omega} \sum_{-\infty}^{\infty} f_m e^{-i\omega m}$,

the contribution of all f_{m+1} can be shown to be

$$\sum_{-\infty}^{\infty} f_m e^{-i\omega m} \left\{\frac{12}{\omega^4}(1-e^{i\omega}) + \frac{i}{\omega}\left[1+\frac{6}{\omega^2}(1+e^{i\omega})\right]\right\}. \qquad (2.5\text{-}6b)$$

In a similar way one obtains the contributions of f'_m and f'_{m+1}:

$$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m} \left\{\frac{-1}{\omega^2}\left[1-\frac{6}{\omega^2}(1-e^{-i\omega})\right] - \frac{2i}{\omega^3}(2+e^{-i\omega})\right\} \qquad (2.5\text{-}6c)$$

and $$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m} \left\{\frac{1}{\omega^2}\left[1-\frac{6}{\omega^2}(1-e^{i\omega})\right] - \frac{2i}{\omega^3}(2+e^{i\omega})\right\}. \qquad (2.5\text{-}6d)$$

Introducing $e^{i\omega} + e^{-i\omega} = 2\cos\omega$ and $e^{i\omega} - e^{-i\omega} = 2i\cdot\sin\omega$, the sum of (2.5-6a) + (2.5-6b) and (2.5-6c) + (2.5-6d) can be written as

$$\sum_{-\infty}^{\infty} f_m e^{-i\omega m}\cdot\left[\frac{24}{\omega^4}(1-\cos\omega)-\frac{12}{\omega^3}\sin\omega\right], \qquad (2.5\text{-}7a)$$

$$i\cdot\sum_{-\infty}^{\infty} f'_m e^{-i\omega m}\left[\frac{12}{\omega^4}\sin\omega - \frac{4}{\omega^3}(2+\cos\omega)\right]. \qquad (2.5\text{-}7b)$$

There remains still f'_m to be expressed by f_m; it can be shown that f'_m is a linear combination of f_j

$$f'_m = -3\sum_{j=1}^{\infty}\alpha_j(f_{m+j} - f_{m-j}) \qquad (2.5\text{-}8)$$

with coefficients $\alpha_j = \alpha^j = (-2+\sqrt{3})^j$. Taking this relation into account,

$$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m}$$ can be transformed into

$$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m} = -3\sum_{m=-\infty}^{\infty} e^{-i\omega m}\sum_{j=1}^{\infty}\alpha_j(f_{m+j}-f_{m-j}) \;;$$

taking advantage of the identity

$$\sum_{m=-\infty}^{\infty} e^{-i\omega m}\,\alpha_j f_{m\pm j} = e^{\pm j\omega}\,\alpha_j\sum_{m=-\infty}^{\infty} f_m \;,$$

which is the frequency equivalent of a "time-shift", and interchanging the sequences of summation over m and j, one obtains

$$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m} = \left[-3\sum_{j=1}^{\infty}\alpha_j(e^{ij\omega}-e^{-ij\omega})\right]\cdot\left[\sum_{m=-\infty}^{\infty} f_m e^{-i\omega m}\right],$$

and with $e^{ij\omega} - e^{-ij\omega} = 2i \cdot \sin(j\omega)$,

$$\sum_{-\infty}^{\infty} f'_m e^{-i\omega m} = -6i\left[\sum_{j=1}^{\infty} \alpha_j \sin(j\omega)\right] \cdot \left[\sum_{m=-\infty}^{\infty} f_m e^{-i\omega m}\right]. \qquad (2.5\text{-}9)$$

Because $|\alpha| = 0.2768 < 1$, the first sum in above equation can be expressed in closed form (Gradshteyn, No. 1.447, p. 40),

$$\sum_{j=1}^{\infty} \alpha^j \sin(j\omega) = \sum_{j=1}^{\infty} \frac{\alpha \sin \omega}{1-2\alpha \cos\omega + \alpha^2} \quad ,$$

which, with $\alpha = -2 + \sqrt{3}$ reduces to

$$\sum_{j=1}^{\infty} \alpha^j \sin(j\omega) = -\frac{1}{2} \frac{\sin \omega}{2 + \cos\omega} \quad , \qquad (2.5\text{-}10)$$

and, with (2.5-7b), (2.5-9), and (2.5-10), the contribution of all f'_m to the spectrum is given by

$$\left(-\frac{36 \sin^2\omega}{\omega^4(2+\cos\omega)} + \frac{12}{\omega^3} \sin\omega\right) \cdot \sum_{-\infty}^{\infty} f_m e^{-i\omega m} \quad .$$

Adding the contribution of f_m from equation (2.5-7a), we obtain for the whole spectrum

$$F(\omega) = \frac{12\ (1-\cos\omega)^2}{\omega^4(2+\cos\omega)} \sum_{-\infty}^{\infty} f_m e^{-i\omega m} \quad ; \qquad (2.5\text{-}11)$$

with the identity

$$\frac{4}{\omega^4}(1-\cos\omega)^2 = \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 \quad ,$$

the Fourier transform of the cubic spline is finally given by

$$F(\omega) = \frac{3}{2+\cos\omega}\left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 \sum_{-\infty}^{\infty} f_m e^{-i\omega m} \quad . \qquad (2.5\text{-}12)$$

This result is very interesting and deserves to be discussed:

Let us first investigate the summation part of $F(\omega)$:

$$F_\Sigma(\omega) = \sum_{-\infty}^{\infty} f_m e^{-i\omega m} \quad . \qquad (2.5\text{-}13)$$

This is precisely the discrete Fourier transform of the infinite data vector $\{f_m\}$, $m = -\infty, \ldots, \infty$ (see e.g., E.O. Brigham, 1974). Therefore, its inverse Fourier transform should again be the data vector:

$$f_\Sigma(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F_\Sigma(\omega) e^{i\omega x}\, d\omega$$

$$= \frac{1}{2\pi}\sum_{-\infty}^{\infty} f_m \int_{-\infty}^{\infty} e^{-i\omega m}\, e^{i\omega x}\, d\omega \,. \qquad (2.5\text{-}14)$$

The integral in the above formula can also be written as

$$\int_{-\infty}^{\infty} e^{-i\omega m} e^{i\omega x}\, d\omega = \int_{-\infty}^{\infty} e^{i\omega(x-m)}\, d\omega$$

$$= \int_{-\infty}^{\infty} \cos\omega(x-m)\, d\omega + i\int_{-\infty}^{\infty} \sin\omega(x-m)\, d\omega \,.$$

The second integral vanishes because of the asymmetry of the integrand. The first integral can be split up into

$$\int_{-\infty}^{\infty} \cos \omega (x-m) d\omega = \sum_{k=-\infty}^{\infty} \int_{(2k-1)\pi}^{(2k+1)\pi} \cos \omega (x-m) dx. \qquad (2.5\text{-}15a)$$

Since $$\int_{(2k-1)\pi}^{(2k+1)\pi} \cos \omega (x-m)\, d\omega = \begin{cases} 2\pi & \text{for } x = m \\ 0 & \text{otherwise} \end{cases},$$

the above integral is nothing else but

$$\int_{-\infty}^{\infty} \cos \omega (x-m) d\omega = 2\pi \, \delta (x-m) \qquad (2.5\text{-}15b)$$

with $\delta(\cdot)$ the Dirac delta distribution. Therefore, $f(x)$ in (2.5-14) reduces to

$$f(x) = \frac{1}{2\pi} \sum_{-\infty}^{\infty} f_m \cdot 2\pi \, \delta (x-m)$$

or

$$f(x) = \sum_{-\infty}^{\infty} f_m \delta (x-m),$$

a sequence of impulses $\{f_m\}$, $m = -\infty, \ldots, \infty$, called an "impulse comb" (the product is to be understood in terms of distribution theory). This concludes the inverse Fourier transform of $F_\Sigma(\omega)$.

The first part of $F(\omega)$, the term

$$S(\omega) = \frac{3}{2 + \cos \omega} \left(\frac{\sin \frac{\omega}{2}}{\frac{\omega}{2}} \right)^4 ,$$

is the Fourier transform of the fundamental cardinal cubic spline; this has been proven before. Since $F(\omega)$ is the product of $S(\omega)$ and $F_\Sigma(\omega)$,

$$F(\omega) = S(\omega) F_\Sigma(\omega) ,$$

we conclude from the convolution theorem that a cardinal cubic spline f(x) corresponds to $F(\omega)$ with f(x) being the result of a convolution of a fundamental cardinal cubic spline S(x) with the data sequence (sequence of impulses),

$$f(x) = S(x) ** \sum_{-\infty}^{\infty} f_m \delta(x-m)$$

$$= \sum_{-\infty}^{\infty} f_m \int_{-\infty}^{\infty} S(t)\, \delta(x-m-t)dt,$$

$$f(x) = \sum_{-\infty}^{\infty} f_m S(x-m) \quad .$$

The above equation is identical with (2.4-4) and so is the circle closed. Now we can also see the more elegant way of deriving the Fourier transform of the cubic spline: find the Fourier transform of the fundamental cardinal cubic spline and multiply it with the discrete Fourier transform of the data sequence. This approach can be found in (H. Sünkel, 1980).

Let us once more consider $F_\Sigma(\omega)$ and assume that $F_\Sigma(\omega) = 0$ for $|\omega| > \pi$, or in other words, let $F_\Sigma(\omega)$ be the Fourier transform of a frequency band limited process (function). Then the sum in equation (2.5-15) reduces to a single element, the integral

$$\int_{-\pi}^{\pi} \cos \omega (x-m)d\omega = 2 \frac{\sin \pi (x-m)}{x-m} \quad ,$$

which, with (2.5-14), leads to the interpolation function

$$f(x) = \sum_{-\infty}^{\infty} f_m \frac{\sin \pi(x-m)}{\pi (x-m)} \quad . \qquad (2.5-16)$$

This function is interesting insofar as it can be shown to be an interpolating spline function of highest possible degree ∞ (I.J. Schoenberg, 1973).

It is infinitely often continuously differentiable, but suffers from localness. It is also remarkable in that it is exactly the function which reproduces an original π- band-limited function sampled at a sampling rate equal to 1. This is the essence of the sampling theorem. Therefore, using (2.5-16) as interpolation function, corresponds to the assumption of an originally band-limited function with highest frequency $\omega = \pi$ (the Nyquist frequency).

The cubic spline, however, shows a somehow different behavior: strictly speaking, its spectrum is not band-limited, buc the Fourier transform of the fundamental cardinal cubic spline

$$\frac{3}{2+\cos\omega}\left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 \qquad (2.5\text{-}17)$$

is such that it practically annihilates all frequencies above $\omega = 2\pi$. This factor is of interest because it varies with the degree of the spline. In the case of a step function, which is the spline of lowest degree --0, this dampening factor degenerates to

$$S_0(\omega)\ \frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}} \quad ; \qquad (2.5\text{-}18)$$

in the case of the function (2.5-16), which is the spline of highest degree -- ∞, this factor degenerates to a window

$$S_\infty(\omega) = \begin{cases} 1 & \text{for } |\omega| \leq \pi \\ 0 & \text{otherwise} \end{cases} . \qquad (2.5\text{-}19)$$

In between these two extrema there is the large family of splines of all possible degrees. A graph of the factor (2.5-17), which is at the same time the Fourier transform of the fundamental cardinal cubic spline, is shown in Fig. 2.6.

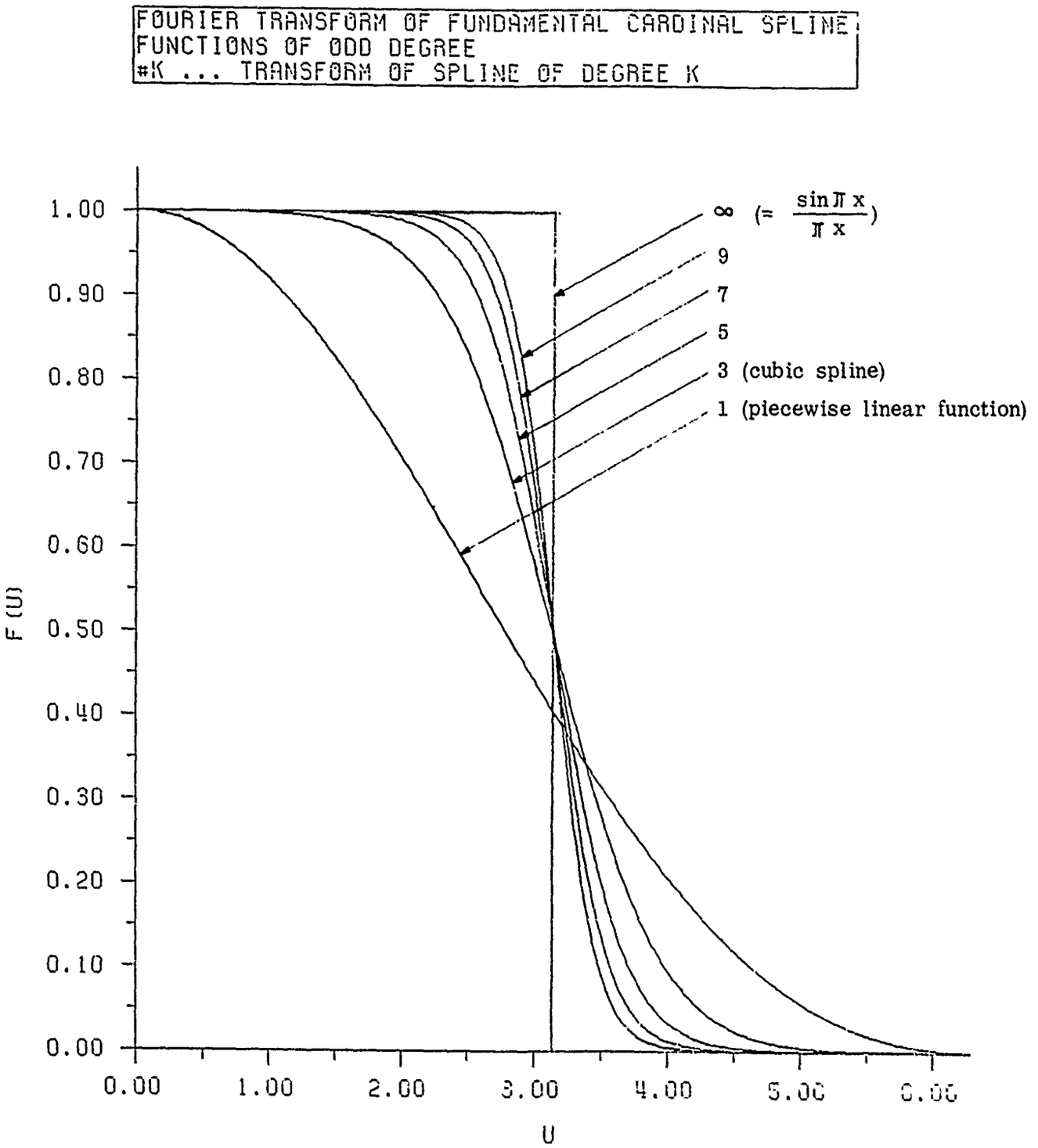


Fig. 2.6 Fourier transforms of a fundamental cardinal cubic spline

The graph shows clearly that the lower frequencies $\omega \leq \pi$ of the cubic spline are dampened relative to the window (2.5-19) of the spline of infinite degree expressed by (2.5-16). This dampening in the lower frequencies $\omega \leq \pi$ is exactly compensated in the higher

frequencies $\omega > \pi$. The reason for this is that the integral of the Fourier transform of the cardinal spline (and all other splines as proven in (H. Sünkel, 1980)) is equal to π,

$$\int_0^\infty \frac{3}{2+\cos\omega} \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}} \right)^4 d\omega = \pi.$$

Consequently, the amount of dampening in the low frequencies $\omega \leq \pi$ is equal to the build-up of frequencies for $\omega > \pi$:

$$\int_0^\infty \left[1 - \frac{3}{2+\cos\omega} \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4\right] d\omega = \int_\pi^\infty \frac{3}{2+\cos\omega} \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 d\omega .$$

Summarizing we can say, that the frequency dampening of a spline of arbitrary degree in the frequency range $|\omega| \leq \pi$ is exactly compensated in the range (π, ∞) , relative to the window (2.5-19). The higher the degree of the spline, the better is the approximation to this window. Therefore, the dampening of the lower frequencies and the build-up of higher frequencies must be caused by the limited degree of continuous differentiability of the spline (a kind of Gibb's phenomenon). For this reason we conclude that the frequencies $\omega > \pi$ are not reliable anymore -- it is more or less frequency noise which is only present in order to compensate the deficiency in the range $\omega \leq \pi$.

To understand this is essential in order to get a better idea of how reliable differentiated splines are.

2.5.1.1 Spectrum of the differentiated cubic spline. The module GSPP is also capable of providing derivates of profiles and surfaces up to and including second order. In order to better understand the reliability of the output, which is a differentiated cubic (bicubic) spline, it is essential to investigate the impact of a differentiation on the spline in the frequency domain.

This time we go the short way and make use of the fact that the cardinal cubic spline is the result of a convolution of a fundamental cardinal cubic spline with an impulse comb of data (the sequence $\{f_m\}$),

$$f(x) = S(x) * \sum_{-\infty}^{\infty} f_m \delta(x-m) \quad . \tag{2.5-20}$$

The differentiated cardinal cubic spline is likewise given by

$$f'(x) = S'(x) * \sum_{-\infty}^{\infty} f_m \delta(x-m) \tag{2.5-21}$$

The corresponding equivalent in the frequency domain is a product of the Fourier transform of $S'(x)$ with the Fourier transform of the impulse comb; the latter is given by equation (2.5-13),

$$F_{\Sigma}(\omega) = \sum_{-\infty}^{\infty} f_m e^{-i\omega m}$$

From (2.5-12) we know the Fourier transform of $S(x)$ to be

$$S(\omega) = \frac{3}{2+\cos\omega} \cdot \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 \quad ,$$

and therefore, $S(x)$ can be obtained from the inverse Fourier transform,

$$S(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{i\omega x} \, d\omega$$

$S'(x)$ as derivative of $S(x)$ with respect to x is then given by

$$S'(x) = \frac{i}{2\pi} \int_{-\infty}^{\infty} S(\omega)\, \omega\, e^{i\omega x} \, d\omega \, , \tag{2.5-22}$$

and its Fourier transform is simply

$$S^{(1)}(\bar{\omega}) = \frac{i}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S(\omega)\, \omega\, e^{i\omega x} e^{-i\bar{\omega}x}\, d\omega\, dx \;;$$

since we know from equation (2.5-15) that

$$\int_{-\infty}^{\infty} e^{\pm i(\omega - \bar{\omega})x}\, dx = 2\pi \cdot \delta(\omega - \bar{\omega}) \,,$$

we obtain

$$S^{(1)}(\bar{\omega}) = i \int_{-\infty}^{\infty} S(\omega)\, \omega\, \delta(\omega - \bar{\omega})\, d\omega \,,$$

which, because of the property of the δ- function, reduces to

$$S^{(1)}(\omega) = i\,\omega\, S(\omega) \,. \qquad (2.5\text{-}23)$$

And finally, the Fourier transform of the differentiated cardinal cubic spline is

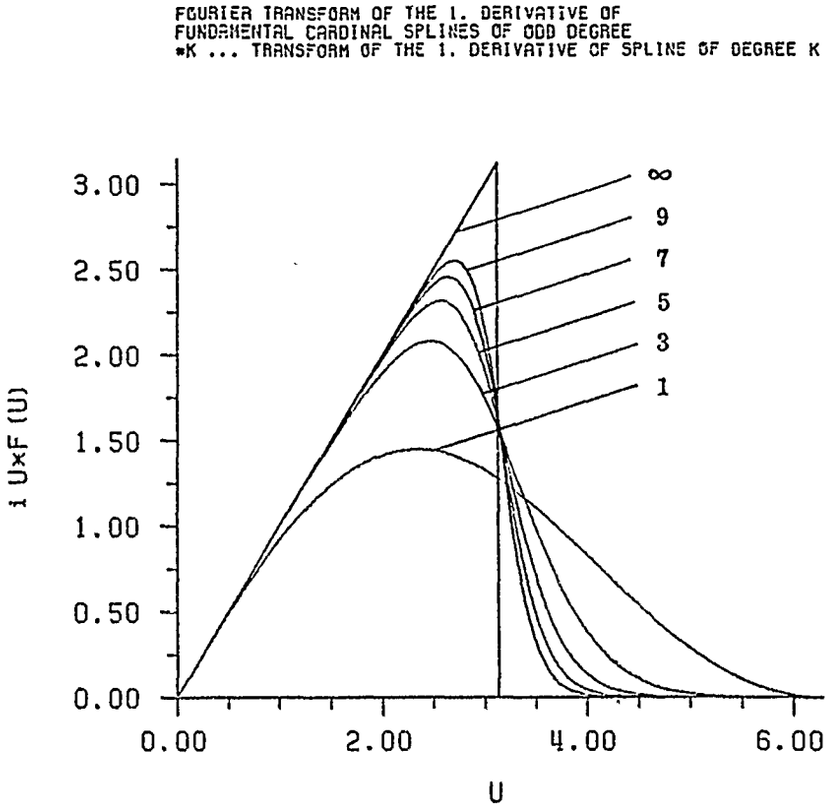
$$F^{(1)}(\omega) = S^{(1)}(\omega) \cdot F_{\Sigma}(\omega) \,,$$

$$\boxed{F^{(1)}(\omega) = i\omega\, \frac{3}{2 + \cos\omega} \left(\frac{\sin\frac{\omega}{2}}{\frac{\omega}{2}}\right)^4 \sum_{-\infty}^{\infty} f_m e^{-i\omega m}} \cdot \qquad (2.5\text{-}24)$$

We recognize the well-known fact that a differentiation dampens the lower ($\omega < 1$) and amplifies the higher ($\omega > 1$) frequencies. For comparison purposes we also give the Fourier transform of the highest possible degree (∞) cardinal spline,

$$F_{\infty}^{(1)}(\omega) = \begin{cases} i\omega \sum_{-\infty}^{\infty} f_m e^{-i\omega m} & \text{for} \quad |\omega| \leq \pi \\ 0 & \text{for} \quad |\omega| > \pi \end{cases} \,.$$

The graphs in Fig. 2.7 give a comparison of both transforms for first and second order derivatives. Notice the shift of the frequency sensitivity



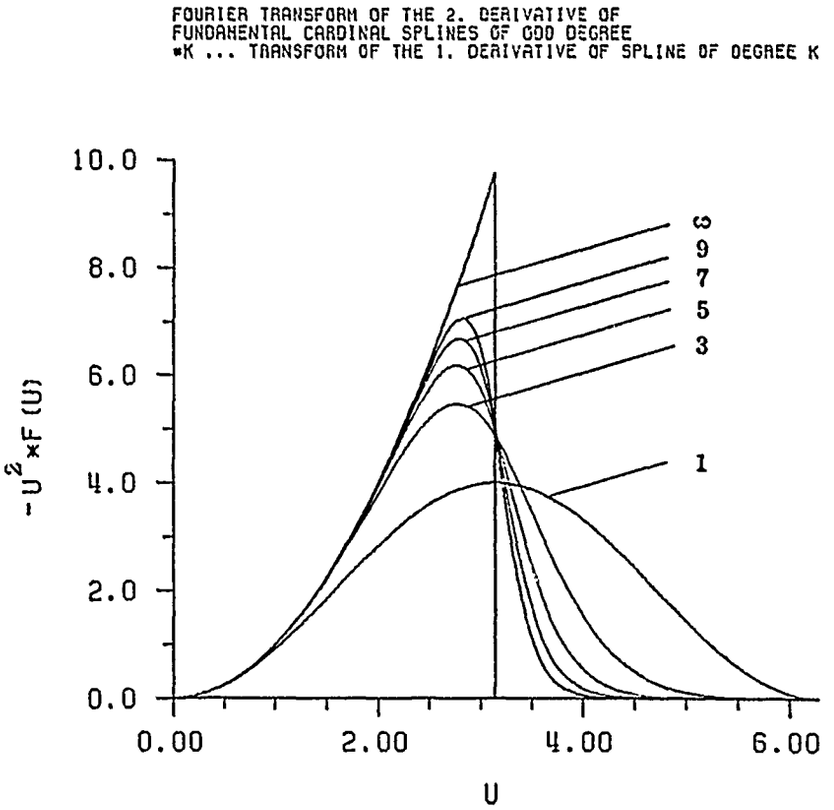


Fig. 2.7 Fourier transforms of spline derivatives

maximum towards $\omega = \pi$ with increasing number of differentiations. What are the consequences of this? Since the frequency resolution becomes worse for higher frequencies, the reliability of spline differentiation decreases with each differentiation and the frequency noise represented by the frequency part $\omega > \pi$ gains more influence on the result of the differentiation.

2.5.2 Spectrum of the bicubic spline. Recalling the defining equation (2.4-13) of the bicubic spline

$$f(x, y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} f_{mn} S(x-m)\ S(y-n) \quad , \tag{2.5-25}$$

with $\{f_{mn}\}$ the infinite data array and $S(x)$ the fundamental cardinal cubic spline, it is obvious that the bicubic spline is a tensor product of cubic splines in x and y. Therefore it is pretty easy to find the two-dimensional Fourier transform

$$F(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i(\omega_x x + \omega_y y)}\, dxdy \ . \tag{2.5-26}$$

Since (2.5-25) is a two-dimensional convolution of a two-dimensional impulse comb with a two-dimensional fundamental cardinal cubic spline

$$f(x, y) = S(x)\ S(y) \ ** \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} f_{mn}\ \delta(x-m)\ \delta(y-n), \tag{2.5-27}$$

the corresponding Fourier transform $F(\omega_x, \omega_y)$ is, according to the convolution theorem, simply a product of the Fourier transform of the 2-D fundamental cardinal cubic spline and the Fourier transform of the 2-D impulse comb.

To find the first is easy, it is just a product of two one-dimensional transforms,

$$S(\omega_x, \omega_y) = S(\omega_x)\ S(\omega_y),$$

$$S(\omega_x, \omega_y) = \frac{9}{(2+\cos\omega_x)(2+\cos\omega_y)} \left(\frac{\sin\frac{\omega_x}{2}\cdot\sin\frac{\omega_y}{2}}{\frac{\omega_x}{2}\quad\frac{\omega_y}{2}}\right)^4 ; \qquad (2.5\text{-}28)$$

the latter is simply the 2-D analogue to (2.5-13),

$$F_\Sigma(\omega_x, \omega_y) = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} f_{mn} e^{-i(\omega_x m + \omega_y n)} . \qquad (2.5\text{-}29)$$

And finally, with (2.5-27), the 2-D Fourier transformation of the cardinal bicubic spline is given by

$$S(\omega_x, \omega_y) = \frac{9}{(2+\cos\omega_x)(2+\cos\omega_y)} \left(\frac{\sin\frac{\omega_x}{2}\cdot\sin\frac{\omega_y}{2}}{\frac{\omega_x}{2}\quad\frac{\omega_y}{2}}\right)^4 \cdot$$

$$\sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} f_{mn} e^{-(\omega_x m + \omega_y n)} . \qquad (2.5\text{-}30)$$

Expressed in terms of the bicubic spline coefficients, the above equation has the form which is analogous to (2.5-5) of

$$S(\omega_x, \omega_y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\sum_{k=0}^{3}\sum_{l=0}^{3} a_{kl}^{(m,n)} e^{-i(\omega_x m + \omega_y n)} \cdot$$

$$\frac{k!\,l!}{(i\omega_x)^{k+1}(i\omega_y)^{l+1}} \cdot \left[1 - e^{-i\omega_x}\sum_{r=0}^{k}\frac{(i\omega_x)^{k-r}}{(k-r)!}\right] \cdot$$

$$\left[1 - e^{-i\omega_y}\sum_{s=0}^{l}\frac{(i\omega_y)^{l-s}}{(l-s)!}\right] . \qquad (2.5\text{-}31)$$

An equivalent expression can also be found in (Bhattacharyya, 1969). All that has been stated so far for the one-dimensional spline, carries over to the two-dimensional one:

There exists a highest possible degree (∞, ∞) 2-D cardinal spline, which is fully analogous to (2.5-16), with the form

$$f(x, y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} f_{mn} \frac{\sin \pi (x-m)}{\pi (x-m)} \cdot \frac{\sin \pi (y-n)}{\pi (y-n)} . \qquad (2.5-32)$$

This function is infinitely often differentiable, but suffers from localness if compared with the bicubic spline. It is the interpolating function which reproduces a band-limited function with highest frequencies $\omega_x = \pi$, $\omega_y = \pi$ at a sampling rate equal to 1. Therefore, using (2.5-32) as interpolation function, corresponds to the assumption of an originally band-limited function; if this was not the case (as usual in almost all practical applications), the interpolating function suffers from aliasing effects.

The bicubic spline's spectrum is, strictly speaking, not band-limited, but the 2-D Fourier transform of the fundamental cardinal bicubic spline (2.5-28) annihilates practically all frequencies above $\omega_x = 2\pi$, $\omega_y = 2\pi$. The frequencies above $\omega = \pi$, however, are caused by the discontinunity of the splines third derivative. It is a kind of Gibb's phenomenon; therefore, these higher frequencies represent more or less only "discontinuity noise", which is exactly compensated in the frequency range $|\omega| \leq \pi$. This compensation causes the lower frequencies to be dampened (compare Fig. 2.7).

2.6 Practical 1-D and 2-D spline routines

2.6.1 The cubic spline routine. In chapters 2.4 and 2.5 a somewhat artificial case has been discussed: splines based on an infinite number of data, uniformly distributed on a grid with constant grid distance equal to 1. The discussion has been performed for this

exotic case because it is considerably simpler to point out the essential features; also the finite spline behaves likewise because of its localness. For applications, however, it is necessary to deal with the general case.

Detailed derivations can be found in a standard text on splines like (Ahlberg et al., 1967). Here, only the very necessary things will be presented.

Let there be given a sequence of abscissas $\{x_m\}$, m = 1, ..., M; $f_m = f(x_m)$. An interpolating cubic spline is a twice continuously differentiable function over the range [a, b] which consists of (M-1) cubic polynomials, each of them defined on an interval $[x_i, x_{i+1}]$,

$$f^{(m)}(x) = \sum_{k=0}^{3} a_k^{(m)} (x-x_m)^k \quad . \qquad (2.6\text{-}1)$$

The coefficients $\{a_k^{(m)}\}$, m=1 . . ., M-1, can be found via the continuity conditions for first or second order derivatives. This leads to a system of M linear equations

$$Af'' = Bf \qquad (2.6\text{-}2a)$$

A is a strongly diagonally dominant tridiagonal symmetric matrix with positive diagonal elements; it follows from well known theorems in matrix algebra, that A is positive definite. The uniqueness of the spline is guaranteed if, in addition to the function values, the second derivatives are given at the end points a and b of the interval [a, b]. Since such kind of boundary informations are hardly ever available, the most natural choice is to assume them to be zero. This corresponds to a linear behavior of the spline at the end points a and b of the interval. The corresponding cubic spline is referred to as a "natural spline". GSPP assumes vanishing 2nd derivatives at the boundary points, and therefore, calculates exclusively natural cubic splines.

The solution of (2.6-2)

$$f'' = A^{-1}Bf, \tag{2.6-2b}$$

can be found very efficiently by a Gauss-Jordan elimination procedure (Späth, 1973; p. 10 ff.). If the system has been solved, it is fairly easy to find the coefficients $\{a_k^{(m)}\}$, $k = 0, \ldots, 3$; $m=1, \ldots, M-1$:

$$\begin{aligned} a_0^{(m)} &= f_m, \\ a_1^{(m)} &= \frac{\Delta f_m}{\Delta x_m} - \frac{1}{6}\Delta x_m(f''_{m+1} + 2f''_m), \\ a_2^{(m)} &= \frac{1}{2}f''_m, \\ a_3^{(m)} &= \frac{1}{6\Delta x_m}(f''_{m+1} - f''_m), \end{aligned} \tag{2.6-3}$$

with $\Delta x_m := x_{m+1} - x_m$ and $\Delta f_m := f_{m+1} - f_m$.

In a similar way (2.6-2a, b) and (2.6-3) can be expressed in terms of 1st derivatives, which will be used in the calculation of bicubic spline coefficients.

2.6.2 The bicubic spline routine. Let a regular rectangular grid consist of $M \cdot N$ gridpoints and let the function values at the grid points be $\{f_{mn}\}$, $m=1, \ldots, M$; $n=1, \ldots, N$. Then a bicubic spline consists of $(M-1)(N-1)$ bicubic polynomials in x and y

$$f^{(m,n)}(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} a_{kl}^{(m,n)}(x-x_m)^k(y-y_n)^l \tag{2.6-4}$$

with coefficients $\{a_{kl}^{(m,n)}\}$ expressed by the product

$$A^{(m,n)} = \{a_{kl}^{(m,n)}\} = H^T(h_x)FH(h_y) \tag{2.6-5a}$$

and the grid spacing matrices $H(h_x)$ and $H(h_y)$,

$$H(h) = \begin{bmatrix} 1 & 0 & -3/h^2 & 2/h^3 \\ 0 & 1 & -2/h & 1/h^2 \\ 0 & 0 & 3/h^2 & -2/h^3 \\ 0 & 0 & -1/h & 1/h^2 \end{bmatrix} , \qquad (2.6\text{-}5b)$$

where h_x = grid distance in x-direction

h_y = grid distance in y-direction .

F is a matrix with data information

$$F = \begin{bmatrix} f_{mn} & q_{mn} & f_{m,n+1} & q_{m,n+1} \\ p_{mn} & r_{mn} & p_{m,n+1} & r_{m,n+1} \\ f_{m+1,n} & q_{m+1,n} & f_{m+1,n+1} & q_{m+1,n+1} \\ p_{m+1,n} & r_{m+1,n} & p_{m+1,n+1} & r_{m+1,n+1} \end{bmatrix} , \qquad (2.6\text{-}5c)$$

where

f_{mn} = function value at the gridpoint (m, n)

p_{mn} = first x-derivative at the gridpoint (m, n)

q_{mn} = first y-derivative at the gridpoint (m, n)

r_{mn} = second xy-derivative at the gridpoint (m, n).

The derivatives $\{p_{mn}\}$, $\{q_{mn}\}$, and $\{r_{mn}\}$ at all gridpoints are determined by continuity conditions of second order derivatives similar to the one-dimensional case. A unique bicubic spline representation, however requires, apart from all function values $\{f_{mn}\}$ at the grid points, the following additional boundary informations:

$$\{p_{1,n}\} , \{p_{M,n}\} , \quad n=1, \ldots, N;$$
$$\{q_{m,1}\} , \{q_{m,N}\} , \quad m=1 , \ldots, M; \qquad (2.6\text{-}5d)$$

$\{r_{mn}\}$, m=1, M; n=1, N.

Since these boundary informations (derivatives) are hardly ever available in practical applications, similarly as in the one-dimensional case, assumptions have to be made concerning their values. The most natural assumption is to assume vanishing second order derivatives along the boundary normal. This is an implicite assumption relative to the boundary data (2.6-5d). The practical determination of the coefficients runs then as follows:

1.) Solve the spline equations (2.6-2) for first order derivatives as unknowns with data vectors $\{f_{mn}\}$, m=1,..., M and vanishing 2nd derivatives at the boundary points, $D_x^2 f(x, y_n)|_{x=x_1} = D_x^2 f(x, y_n)|_{x=x_M} = 0$, for all "columns" n=1, ..., N obtaining so all 1. derivatives in x-direction $\{p_{mn}\}$, m=1, ..., M; n=1, ..., N.

2.) Interchange the role of x and y: Solve the spline equations (2.6-2) for first order derivatives as unknowns with data vectors $\{f_{mn}\}$, n=1, ..., N and vanishing 2 nd derivatives at the boundary points, $D_y^2 f(x_m, y)|_{y=y_1} = D_y^2 f(x_m, y)|_{y=y_N} = 0$, for all "rows" m=1, ..., M obtaining so all first derivatives in y-direction $\{q_{mn}\}$, m=1, ..., M; n=1, ..., N.

3.) Replace "f" by "q" and solve according point (1) or, which is equivalent, replace "f" by "r" and solve according point (2). The result is all second order mixed derivatives $\{r_{mn}\}$, m=1, ..., M; n=1, ..., N.

With these values obtained, the bicubic spline is completely defined. (The coefficients $\{a_{kl}^{(m,n)}\}$ depend only on $\{f_{mn}, p_{mn}, q_{mn}, r_{mn}\}$, m=1, ..., M; n=1, ..., N and on the grid spacing h_x and h_y.)

Therefore, it is necessary to have this set of defining values, the quadrupel $(f, p, q, r)_{mn}$ at each grid point, available in order to define a spline uniquely. Generally speaking, also the grid distances h_x and h_y can change with m, $h_x = h_x(m)$, h_y can change with n, $h_y = h_y(n)$. However, GSPP always assumes h_x and h_y to be constant (they need not necessarily be equal). This simplifies computations quite considerably: the grid information matrices become constant; therefore, it is possible to perform all calculation normalized, with a constant grid spacing in x and y-direction equal to 1. Of course, the calculated derivatives have to be interpreted accordingly -- they refer to grid spacing equal to 1 and have to be scaled by h_x and h_y later on in order to obtain real values. The gain is two-fold: firstly, all derivative calculations (solutions of the spline equations) are highly stabilized, and secondly, the calculation of the product (2.6-5a) -- which, in GSPP, is not formulated in terms of two matrix products but is programmed explicitly -- gains in calculation speed because a number of divisions (or multiplications)is avoided . This sounds trivial but is very essential when the grid is extended and if a huge number of interpolations has to be performed.

2.6.3 Interpolation/differentiation of splines. Interpolation with cubic and bicubic splines is really trivial as soon as the defining values are available -- function values and 1st, or 2nd order derivatives at the knots $\{x_m\}$, m=1, ..., M in the one-dimensional case, and the quadruple $\{f, p, q, r\}_{mn}$, m=1, ..., M; n=1, ..., N in the two-dimensional case. Therefore, an interpolation of a function value at a point x (x, y) requires the following steps to be performed:

1) find the interval number m (grid element numbers m, n) to which the point x (x, y) belongs;

2) calculate the cubic spline coefficients using (2.6-3) (bicubic spline coefficients using (2.6-5) ;

3) perform the product (2.6-1), for the cubic, (2.6-4) for the bicubic spline.

Computation efficiency can be gained by performing the products (2.6-1) and (2.6-4) not blindly, but by reducing the number of operations involved to a minimum like

$$f(x) = a_0 + \bar{x}\,(a_1 + \bar{x}\,(a_2 + \bar{x}\,a_3)), \quad \bar{x} = x - x_m \tag{2.6-6a}$$

for the cubic spline, and

$$\begin{aligned} f(x) = {} & a_{00} + \bar{y}\,(a_{01} + \bar{y}\,(a_{02} + \bar{y}\,a_{03})) \\ & + \bar{x}\,((a_{10} + \bar{y}\,(a_{11} + \bar{y}\,(a_{12} + \bar{y}\,a_{13}))) \\ & + \bar{x}\,((a_{20} + \bar{y}\,(a_{21} + \bar{y}\,(a_{22} + \bar{y}\,a_{23}))) \\ & + \bar{x}\,(a_{30} + \bar{y}\,(a_{31} + \bar{y}\,(a_{32} + \bar{y}\,a_{33}))))), \quad \bar{x} = x - x_m, \ \bar{y} = y - y_n \end{aligned} \tag{2.6-7a}$$

for the bicubic spline. (This arrangement reduces the number of multiplications from originally at least 28 to 15 and keeps the number of additions constant.) If the coefficients refer to grid spacing 1 (normalized coefficients), then $\bar{x} = (x - x_m)/h_x$ and $\bar{y} = (y - y_n)/h_y$.

Any derivative $0 \le \alpha \le 3$ of the cubic spline (2.6-1) can be expressed by

$$D_x^{\alpha} f(x) = \alpha! \sum_{k=\alpha}^{3} a_k \binom{k}{\alpha} \bar{x}^{k-\alpha}, \quad 0 \le \alpha \le 3; \tag{2.6-8}$$

however, it is more efficient to explicitly write down the derivatives:

$$\begin{aligned} f'(x) &= a_1 + \bar{x}\,(2a_2 + \bar{x}\cdot 3a_3), \\ f''(x) &= 2a_2 + 6a_3\bar{x}, \\ f'''(x) &= 6a_3 \end{aligned} \tag{2.6-6b}$$

Similarly, any derivative $0 \le \alpha_1 \le 3$, $0 \le \alpha_2 \le 3$ of the bicubic spline (2.6-4) can be expressed by

$$D^{\alpha} f(x,y) = \frac{\partial^{|\alpha|} f(x,y)}{\partial x^{\alpha_1} \partial y^{\alpha_2}} = \alpha_1!\,\alpha_2! \sum_{k=\alpha_1}^{3} \sum_{l=\alpha_2}^{3} a_{kl} \binom{k}{\alpha_1} \binom{l}{\alpha_2} \bar{x}^{k-\alpha_1} \bar{y}^{l-\alpha_2} \qquad (2.6\text{-}9)$$

with the double-index $\alpha := (\alpha_1, \alpha_2)$, $|\alpha| = \alpha_1 + \alpha_2$, $0 \le \alpha_1, \alpha_2 \le 3$. This compressed expression, however, is far from optimal as far as CPU-time saving is concerned. Therefore, all partial derivatives should rather be used in the following forms (see also (Sünkel, 1980)):

$$\begin{aligned} f_x(x,y) = {} & a_{10} + \bar{y}\,(a_{11} + \bar{y}\,(a_{12} + \bar{y}\,a_{13})) \\ & + \bar{x}\,(2(a_{20} + \bar{y}(a_{21} + \bar{y}\,(a_{22} + \bar{y}\,a_{23}))) \\ & + \bar{x}\cdot 3\,(a_{30} + \bar{y}\,(a_{31} + \bar{y}\,(a_{32} + \bar{y}\,a_{33})))), \end{aligned} \qquad (2.6\text{-}7b)$$

$$\begin{aligned} f_y(x,y) = {} & a_{01} + \bar{x}\,(a_{11} + \bar{x}\,(a_{21} + \bar{x}\,a_{31})) \\ & + \bar{y}\,(2(a_{02} + \bar{x}\,(a_{12} + \bar{x}\,(a_{22} + \bar{x}\,a_{32}))) \\ & + \bar{y}\cdot 3\,(a_{03} + \bar{x}\,(a_{13} + \bar{x}\,(a_{23} + \bar{x}\,a_{33})))), \end{aligned}$$

$$\begin{aligned} f_{xx}(x,y) = {} & 2(a_{20} + \bar{y}\,(a_{21} + \bar{y}\,(a_{22} + \bar{y}\,a_{23})) \\ & + \bar{x}\cdot 3(a_{30} + \bar{y}\,(a_{31} + \bar{y}\,(32 + \bar{y}\,a_{33})))), \end{aligned}$$

$$\begin{aligned} f_{xy}(x,y) = {} & a_{11} + \bar{y}\,(2a_{12} + \bar{y}\cdot 3a_{13}) \\ & + \bar{x}\,(2(a_{21} + \bar{y}\,(2a_{22} + \bar{y}\cdot 3a_{23})) \\ & + \bar{x}\cdot 3\,(a_{31} + \bar{y}\,(2a_{32} + \bar{y}\cdot 3a_{33}))), \end{aligned}$$

$$\begin{aligned} f_{yy}(x,y) = {} & 2(a_{02} + \bar{x}\,(a_{12} + \bar{x}\,(a_{22} + \bar{x}\,a_{32})) \\ & + \bar{y}\cdot 3\,(a_{03} + \bar{x}\,(a_{13} + \bar{x}(a_{23} + \bar{x}\,a_{33})))), \end{aligned}$$

$$f_{xxx}(x,y) = 6(a_{30} + \bar{y}\,(a_{31} + \bar{y}\,(a_{32} + \bar{y}\,a_{33}))),$$

$$f_{xxy}(x,y) = 2(a_{21} + \bar{y}\,(2a_{22} + \bar{y}\cdot 3a_{23}) + \bar{x}\cdot 3(a_{31} + \bar{y}\,(2a_{32} + \bar{y}\cdot 3a_{33}))),$$

$$f_{xyy}(x,y) = 2(a_{12} + \bar{x}\,(2a_{22} + \bar{x}\cdot 3a_{32}) + \bar{y}\cdot 3\,(a_{13} + \bar{x}\,(2a_{23} + \bar{x}\cdot 3a_{33}))),$$

$$f_{yyy}(x,y) = 6(a_{03} + \bar{x}\,(a_{13} + \bar{x}\,(a_{23} + \bar{x}\,a_{33}))),$$

$$f_{xxxy}(x,y) = 6(a_{31} + \bar{y}\,(2a_{32} + \bar{y}\cdot 3a_{33})),$$

$$f_{xxyy}(x,y) = 4(a_{22} + \bar{y}\cdot 3a_{23} + \bar{x}\cdot 3\,(a_{32} + \bar{y}\cdot 3a_{33})),$$

$$f_{xyyy}(x,y) = 6(a_{13} + \bar{x}\,(2a_{23} + \bar{x}\cdot 3a_{33})),$$

$$f_{xxxyy}(x,y) = 12\,(a_{32} + \bar{y}\cdot 3a_{33}),$$

$$f_{xxyyy}(x,y) = 12\,(a_{23} + \bar{x}\cdot 3a_{33}),$$

$$f_{xxxyyy}(x,y) = 36\,a_{33}\,.$$

GSPP is capable of providing plots of derivatives $0 \le \alpha_1\,,\ \alpha_2 \le 2$.

2.7 Contour finding

Let the prediction of function values at a regular rectangular grid be done and assume that the spline routines have generated the quadruple $\{f,p,q,r\}_{mn}$, m=1, ..., M; n=1, ..., N of bicubic spline defining values. In other words, let a bicubic spline surface be given such that on each grid element $R_{mn}\,[x_m,\ x_{m+1};\ y_n,\ y_{n+1}]$ the smooth interpolation function given by (2.6-4). The goal is to find the contour f(x, y) = constant = c.

Consider just one single grid element R_{mn} with the bicubic polynomial (2.6-4) as interpolating element. Then it is obvious that at least one contour exists if the condition

$$\min_{(x,y)\in R_{mn}} f(x,y) \;<\; c \;<\; \max_{(x,y)\in R_{mn}} f(x,y)$$

is fulfilled. In the case of one-sided equality the contour degenerates

to a point, in the case of two-sided equality the bicubic element is flat and horizontal, all coefficients $\{a_{kl}\}$, $k,l = 0, \ldots, 3$ are zero apart from a_{00} (which, naturally, can also vanish), and no contour exists.

The decision process of existence or non-existence of a contour (or more contours) is only the beginning of the long procedure of actually finding them. Remember that we have to find the solution(s) of the equation

$$f(x,y) = \sum_{k=0}^{3} \sum_{l=0}^{3} a_{kl} x^k y^l = c = \text{const.}, \qquad (2.7\text{-}1)$$

a slightly simplified form of (2.6-4). In one dimension this corresponds to the solution of

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 = c,$$

which is nothing else but finding the zeros (≤ 3) of

$$(a_0 - c) + a_1 x + a_2 x^2 + a_3 x^3 = 0 \qquad (2.7\text{-}2)$$

with almost arbitrary coefficients. Even in such a simple case, a simple solution usually does not exist. The approximation methods used for the solution of above cubic equation essentially consist of finding the intersection of the horizontal line $f_1(x) = c$ and a line segment of (2.7-2) which represents the behavior of this cubic polynomial in the neighborhood of the zero being considered. In other words, the zeros can be found by approximating the cubic polynomial (2.7-2) by a continuous and piecewise linear function, and intersecting this approximation function with the line $f_1(x) = c$. It is obvious that each line segment of the piecewise linear function has either 1 intersection or none (Fig. 2.7.1)

In view of these facts it should be clear that direct contour-finding with the bicubic function (2.7-1) is hopeless. There is, however, a way to overcome this problem, similar to that discussed above:

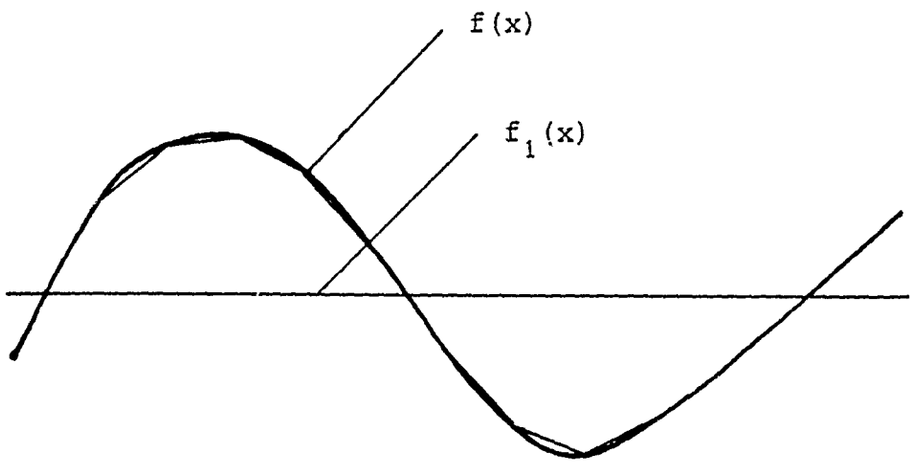


Fig. 2.7.1 Intersection of f(x) with f_1 (x)

By approximating the bicubic function (2.7-1) by a number of simple elements which allow for a simple contour-finding.

The simplest such approximation function is obviously a plane. A plane either has 1 intersection with another (horizontal) plane or has no intersection; moreover, the intersection is a straight line. A plane is uniquely defined by 3 parameters which can be taken as the function values at 3 not coinciding points. Therefore, one would conclude that piecewise linear (flat) triangular elements are the obvious choice for the approximation of any continuous two-dimensional function analogous to the one-dimensional case.

A linear triangular element is defined by the equation

$$f(x, y) = b_0 + b_1 x + b_2 y \qquad (2.7\text{-}3)$$

and, intersected with the horizontal plane

$$f_1(x, y) = c,$$

gives an intersection

$$y(x) = \frac{1}{b_2}(c - b_0 - b_1 x) \qquad (2.7\text{-}4)$$

which is obviously the equation of a straight line. Since the element is flat, only two boundary lines of the triangular element can have an intersection with (2.7-4) or no boundary line has an intersection. To find these intersections is indeed very simple: first it is necessary to check if

$$\min_{i=1,2,3} f_i < c < \max_{i=1,2,3} f_i \quad ,$$

where f_i , $i = 1, 2, 3$ are the function values at the 3 corners of the triangle; the second step consists in the actual calculation of the 2 intersection points. The straight line connecting these two intersection points is part of a contour. The full contour is then the continuous and piecewise linear line consisting of the above described line segments (Fig. 2.7.2).

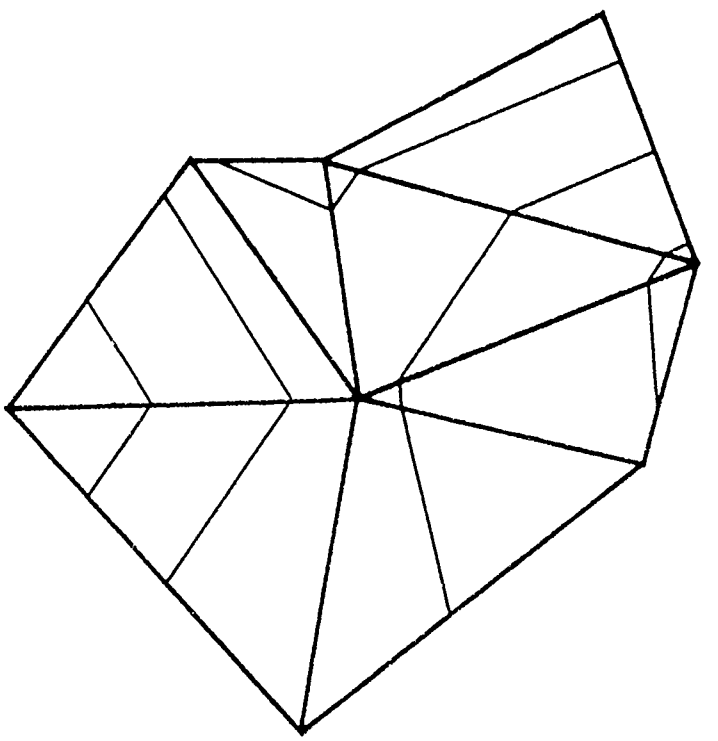


Fig. 2.7.2 Contours on piecewise linear triangular elements

This is in principle the contour-finding procedure frequently used in connection with detailed digital terrain models: sampled terrain heights and other kinds of structure informations like rivers, lakes, roads, dams, artificial structures are taken into account, a dense "triangulation is generated", and the terrain heights at the corner points of the triangles are either known or are predicted. It should be pointed out that all this sounds very simple, but is extremely complicated. The enormous problem consists primarily of analyzing how the human intellect "produces" an image on the basis of data and secondly, to "translate" this stream of logical operations in a programming language.

We are not going to discuss this method in detail because GSPP has been designed for another possible method of contour finding.

Recall that the bicubic polynomial (2.7-1), as part of the bicubic spline surface, is defined on a rectangle. Therefore it is quite natural to approximate the surface not by flat triangular elements, but by simple elements also defined on a rectangle. The simplest element defined on a rectangle is a function with 4 parameters which can be determined from the function values at the 4 corners of the rectangle. Such a function, in general, is no plane anymore; it is a hyperbolic paraboloid (saddle surface) with the equation

$$f(x, y) = b_0 + b_1x + b_2y + b_3xy \qquad (2.7\text{-}5a)$$

this function is bilinear, which can be seen more easily in the equivalent form

$$f(x, y) = (c_0 + c_1x)(d_0 + d_1y) \qquad (2.7\text{-}5b)$$

with

$$\begin{aligned} c_0d_0 &= b_0, \\ c_1d_0 &= b_1, \\ c_0d_1 &= b_2, \\ c_1d_1 &= b_3, \end{aligned}$$

or in the form

$$f(x,y) = \sum_{k=0}^{1} \sum_{l=0}^{1} b_{kl} x^k y^l \qquad (2.7\text{-}5c)$$

with

$$b_{00} = b_0,$$
$$b_{10} = b_1,$$
$$b_{01} = b_2,$$
$$b_{11} = b_3.$$

Bilinear means that the function is linear along the coordinate lines x = const. and along the coordinate lines y = const. In any other direction the function, in general, not linear (Fig. 2.7.3).

This bilinear element plays a central role in contouring based on rectangles and a thorough discussion is essential for an understanding of the whole contouring logics. We will, therefore, investigate its properties in detail.

Let us first introduce a new coordinate system (x', y') parallel to the old system (x, y) with origin coordinates (x_0, y_0) in order to eliminate the linear terms in (2.7-5a),

$$x = x' + x_0, \quad y = y' + y_0 . \qquad (2.7\text{-}6)$$

Then the intersection between the bilinear element (2.7-5a) and a horizontal plane $f_1(x, y) = c$ = constant assumes the form

$$\begin{aligned} c &= b_0 + b_1(x' + x_0) + b_2(y' + y_0) + b_3(x' + x_0)(y' + y_0) \\ &= (b_0 + b_1 x_0 + b_2 y_0 + b_3 x_0 y_0) + (b_1 + b_3 y_0)x' \\ &\quad + (b_2 + b_3 x_0)y' + b_3 x' y' ; \end{aligned} \qquad (2.7\text{-}7)$$

the linear terms (x', y') vanish if $b_1 + b_3 y_0 = 0$ and $b_2 + b_3 x_0 = 0$; these conditions provide the coordinates of the origin of the new coordinate system (x', y')

$$x_0 = -\frac{b_2}{b_3}, \quad y_0 = -\frac{b_1}{b_3} . \qquad (2.7\text{-}8)$$

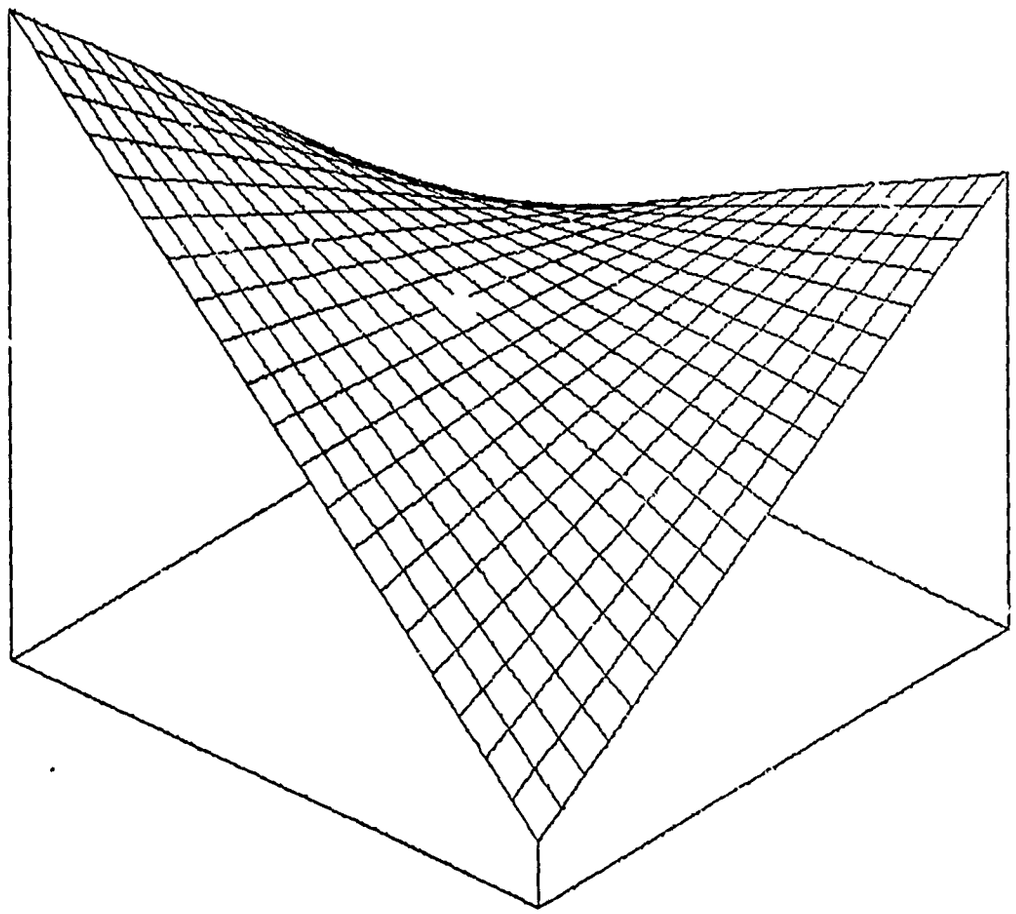


Figure 2.7.3 : A bilinear element

With these values equation (2.7-7) takes on the simple form

$$x'y' = \frac{1}{b_3}\left(c - b_0 + \frac{b_1 b_2}{b_3}\right) . \qquad (2.7\text{-}9)$$

Since the right hand side of the equality sign is constant, the line of intersection has the equation $x'y' = \text{constant}$. But this is exactly the equation of a hyperbola with asymptotic lines parallel to the coordinate lines (because $x' = \text{const.}/y'$ and $y' = \text{const.}/x'$). Therefore, we know also the axes of the hyperbola: they are mutually orthogonal and span an angle of 45° with the coordinate lines. In order to prove this we introduce a new coordinate system $(\bar{x}, \bar{y})$ rotated by an angle α :

$$\begin{aligned} x' &= \bar{x} \cos\alpha - \bar{y} \sin\alpha \\ y' &= \bar{x} \sin\alpha + \bar{y} \cos\alpha \,. \end{aligned} \tag{2.7-10}$$

With this coordinate transformation, the product $x'y'$ assumes the form

$$\begin{aligned} x'y' &= (\bar{x}^2 - \bar{y}^2) \cos\alpha \sin\alpha + \bar{x}\bar{y}(\cos^2\alpha - \sin^2\alpha) \\ &= (\bar{x}^2 - \bar{y}^2)\frac{1}{2}\sin 2\alpha + \bar{x}\bar{y}\cos 2\alpha = \text{const.} \end{aligned}$$

The above equation becomes purely quadratic if $\cos 2\alpha = 0$ which corresponds to $\alpha = \frac{\pi}{4}$ proving the statement made above. So we finally obtain the equation of the contour

$$\bar{x}^2 - \bar{y}^2 = \frac{2}{b_3}\left(c - b_0 + \frac{b_1 b_2}{b_3}\right) . \tag{2.7-11}$$

This purely quadratic expression is the mid-point equation of a hyperbola. It refers to a coordinate system $(\bar{x}, \bar{y})$ which is shifted relative to the original system (x, y) by (x_0, y_0) having values (2.7-8), and rotated by an angle of 45°. The axes of the hyperbola coincide with the new coordinate axis ($\bar{x} = 0$, $\bar{y} = 0$) and span therefore the same angle of 45° with the original system. The hyperbolas asymptotes are parallel to the original coordinate lines (Fig. 2.7.4).

After having pointed out all relevant facts concerning the line of intersection, we come back now to the bilinear element. The bilinear element is linear along each coordinate line; therefore, it is also linear along the 4 boundary lines of the rectangle. Each linear function can be intersected at most once by another linear function, and, for the same reason, can there be at most one contour intersection point on

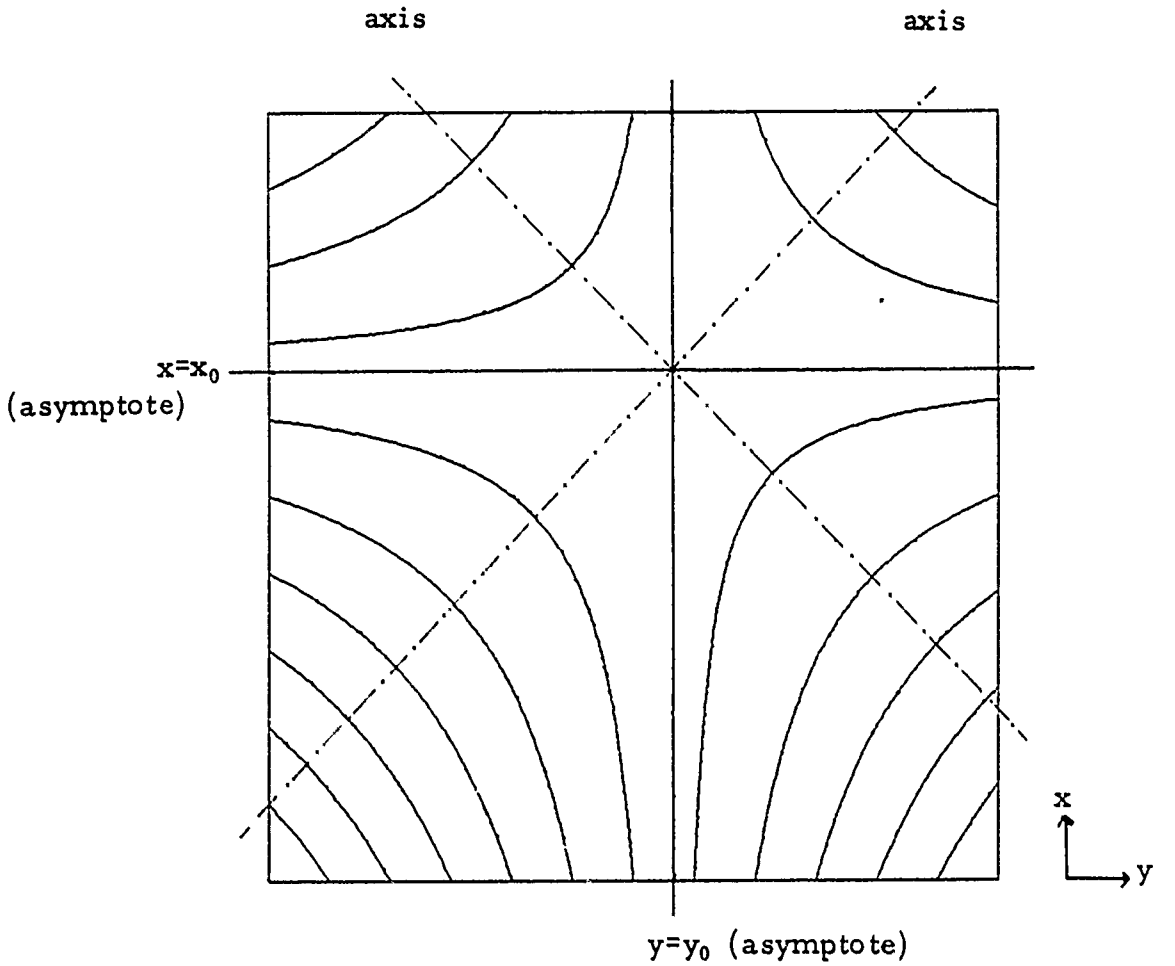


Figure 2.7.4 Contours of a bilinear element

each of the 4 boundary lines of the rectangle. Before we discuss this maximum case we state that the other two possibilities are 2 intersection points or no intersection at all. In the case of only two intersection points it is clear that both points belong to one and the same hyperbola (because the asymptotic lines are parallel to the original coordinate lines), and the connection between the two points is clear. The way of making the connection between intersection points is not so evident (Schumaker, 1976, p. 249 ff). Is there a simple and unique answer to how

the 4 intersection points in Fig. 2.7.5 have to be connected?

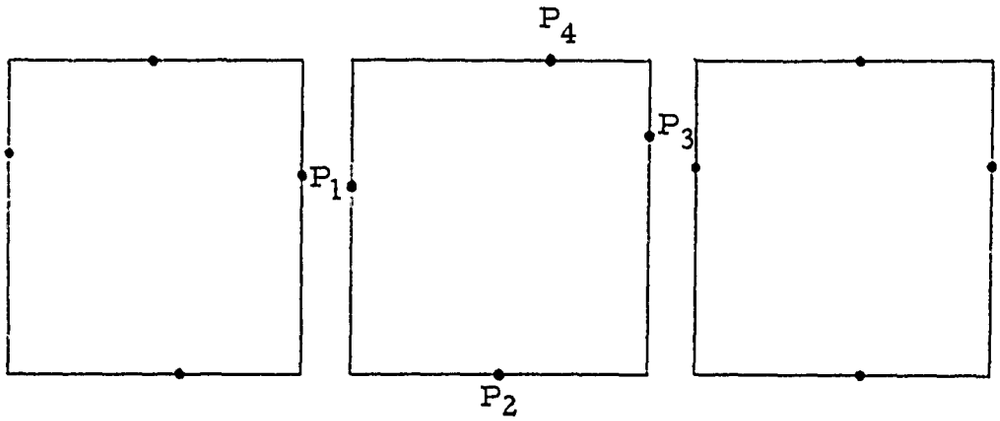


Figure 2.7.5: Bilinear elements -- contour/boundary intersection points .

Yes, there is a simple answer (Sünkel, 1977a). The clue is that the asymptotic lines of the hyperbolas are parallel to the coordinate lines. Therefore, the position of 1 out of the 4 intersection points relative to the origin (x_0, y_0) of the new coordinate system determines the way of point connection uniquely: In Fig. 2.7.5 the point P_1 is below the asymptotic line $x = x_0$; therefore, it can only be connected with P_2 whose coordinate y is smaller than y_0 ; since the hyperbolas are symmetric relative to the origin, the connection of P_3 with P_4 is automatically fixed. Consequently, the simple calculations of the origin (2.7-8) provide sufficient information about the way of point connection (Fig. 2.7.6).

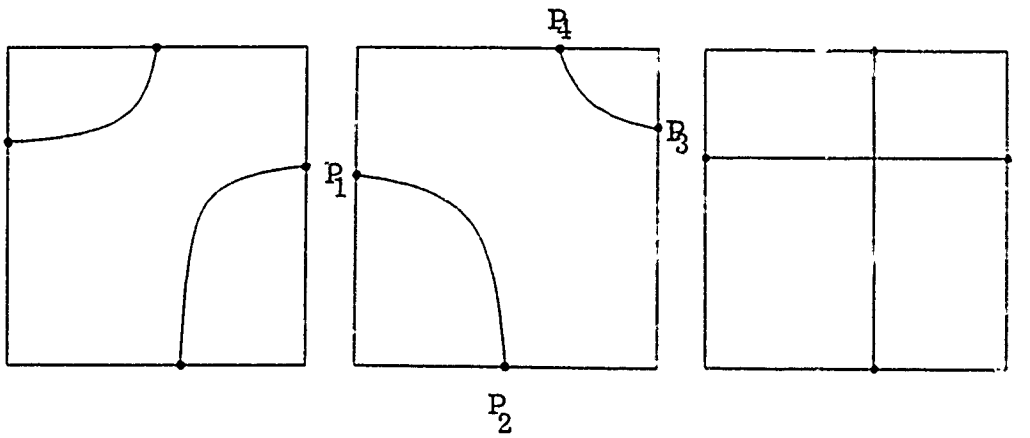


Figure 2.7.6: Bilinear elements -- connection of contour/ boundary intersection points.

The coordinates of the hyperbolas origin are valuable also for another reason: they tell if 4 intersection points are possible (2 hyperbolas) or if just 2 intersection points (1 hyperbola) exists within the square; if the origin is located outside the square, then at most one hyperbola exists, if it is located within the square, two hyperbolas are possible. There is an exceptional case which should also be mentioned: the case of hyperbolas degenerating into the asymptotic lines; this happens if the x-coordinates of one pair of points (and the y-coordinates of the other pair of points) coincide -- such lines are known as saddle lines (see Fig. 2.7.6).

All the considerations made above about the intersection curve(s) and the way of intersection-point connection are simple but essential for the logics of contour finding.

Let us go back to the coefficients $\{b_k\}$, $k = 0, \ldots, 3$, in equation (2.7-5a). They can easily be determined as linear combinations

of the function values at the 4 corners of the square:

$$\begin{aligned} b_0 &= f_{00} \\ b_1 &= f_{10} - f_{00} \\ b_2 &= f_{01} - f_{00} \\ b_3 &= (f_{11} - f_{10}) - (f_{01} - f_{00}) \end{aligned} \qquad (2.7\text{-}12)$$

with $f_{kl} = f(x = k,\ y = l)$. Here it has been assumed that the square on which the particular bilinear element is defined, has unit length. This saves a number of calculations (divisions or multiplications) in GSPP.

Summarizing, the process of contour-finding for a bicubic spline surface consists basically of the following steps:

1) Interpolate the bicubic spline at the grid points of a rectangular array which is a proper subdivision of the original grid on which the spline has been defined (Fig. 2.7.7). The function approximating the bicubic spline is defined by the array of function values at these subgrid points. The function is continuous and consists of bilinear elements, each defined on one individual subgrid element. Store this approximating function.
2) Start with the lowest possible contour value and search for a contour intersection point along the boundary rectangle of the whole subgrid -- contours intersecting the boundary are open (non-periodic) with respect to the contouring area in consideration. Having found the first intersection points, there are two alternatives to find the next one in this element: either by first comparing the contour value with the other 4 remaining contour values, singling out the possible line(s) and calculating the coordinates of the intersection point(s), or to take advantage of the relation between the hyperbolas and their asymptotic lines. Since the origin coordinates

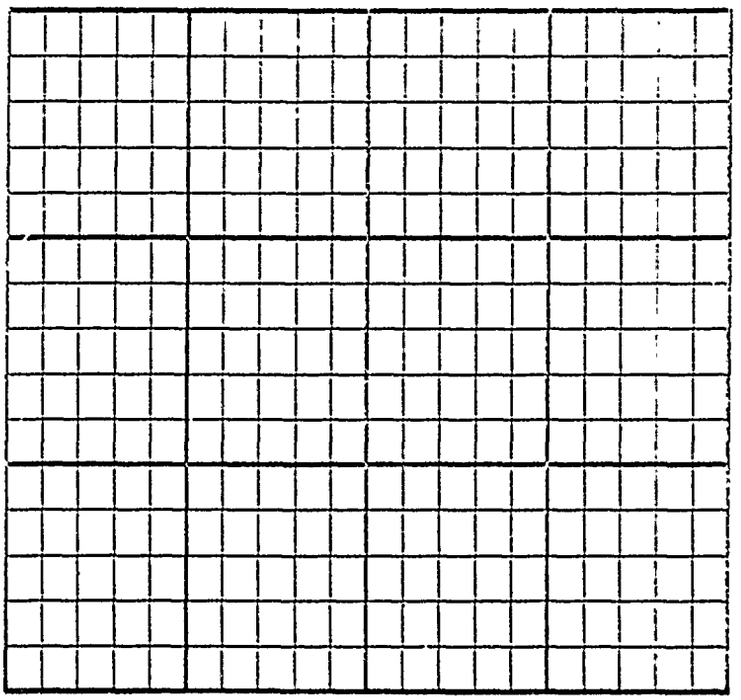


Figure 2.7.7: Grid (4x5) and subgrid (16x21)

are necessary anyway for the decision of how many hyperbolas are possible and for the point connection, a hybrid solution has been chosen. This intersection point searching continues like a domino from one element to the next until the contour leaves one of the four boundaries. A certain integer array associated with the array of bilinear elements "remembers" the position of all intersection points relative to the boundary rectangle of a

particular element, and "directs" the program part which is responsible for the calculation of the contour intersection points. This is necessary in order to avoid a multiple calculation (and plotting) of contours.

After all non-periodic contours (for a particular contour value) have been found, the program searches for periodic contours (contours which do not cross the boundary).

3) Plotting of the contour. Repeat steps (2) and (3) until the contour value exceeds the maximum function value of the surface.

These are the basic steps which are necessary to find contours fo a bicubic spline surface. However, there are and/or can be a number of secondary procedures involved which make the steps (2) and (3) rather complicated. In the sequel, three intermediate procedures are described which GSPP is capable of performing.

2.8 Optional contour procedures

We start here with the most commonly used procedure, the contour smoothing.

2.8.1 Contour smoothing. Theoretically, a contour smoothing would not be necessary if the hyperbolas of the last section (contours of a bilinear element) could be sampled at a sufficiently high rate. This, however, is relatively expensive in terms of CPU-time. Therefore, it has been decided to take only 2 points of a single hyperbola into account, its 2 intersection points with the boundary of the square on which the bilinear element is defined. The actual (non-normalized) size of the square (subgrid distance) is chosen as approximately 2mm by GSPP and can be changed within certain bounds by the user. However, consecutive contour points will have a mutual distance of up to approximately $\sqrt{2}$ * subgrid distance ($\approx$ 3.5 mm in this case). A linear connection of these contour points might give a too rough picture.

Therefore, it is either necessary to decrease the subgrid distance or to smooth the contours. GSPP will always smooth the contours unless the smoothing is suppressed by the user. The interpolation function used for the smooth representation of the contours is a partitioned and overlapping cubic parameter spline.

A cubic parameter spline is a cubic spline whose coordinates x and y depend on a parameter which is here taken to be an approximation of the arc lengths of the contour,

$$x = x(s)$$
$$y = y(s).$$

Therefore, both $x = x(s)$ and $y = y(s)$ are cubic splines depending on the parameter s. In GSPP, the arc length is taken as the accumulated length of the piecewise linear function determined by the originally calculated contour points.

The number of points for a single contour may become quite large (a couple of hundred); moreover, the points are by no means equally spaced. These circumstances can cause instabilities in the spline algorithms. In order to avoid this it has been decided to first "clean-up" the crude vector of contour points which is to be understood as an elimination of all contour points whose mutual distance is smaller than some percentage of the subgrid distance (assumed to be 25% but can be changed by the user within some bounds). After this clean-up process, the vector of contour points is subdivided into a number of overlapping vectors (GSPP: 25 points/vector, 6 points overlapping). The overlapping has been chosen in order to preserve the smooth transition from one part of the contour to the next part to such a degree that it is not possible to detect the transition by visual means; a 6-point overlapping fulfills this requirement; the number 25 has been chosen for reasons of stability.

After the calculation of the spline parameters, further contour points are interpolated such that the overall mutual distance of the

contour points varies only between narrow bounds. Fig. 2.8.1 gives a comparison between smoothed and non-smoothed contours.

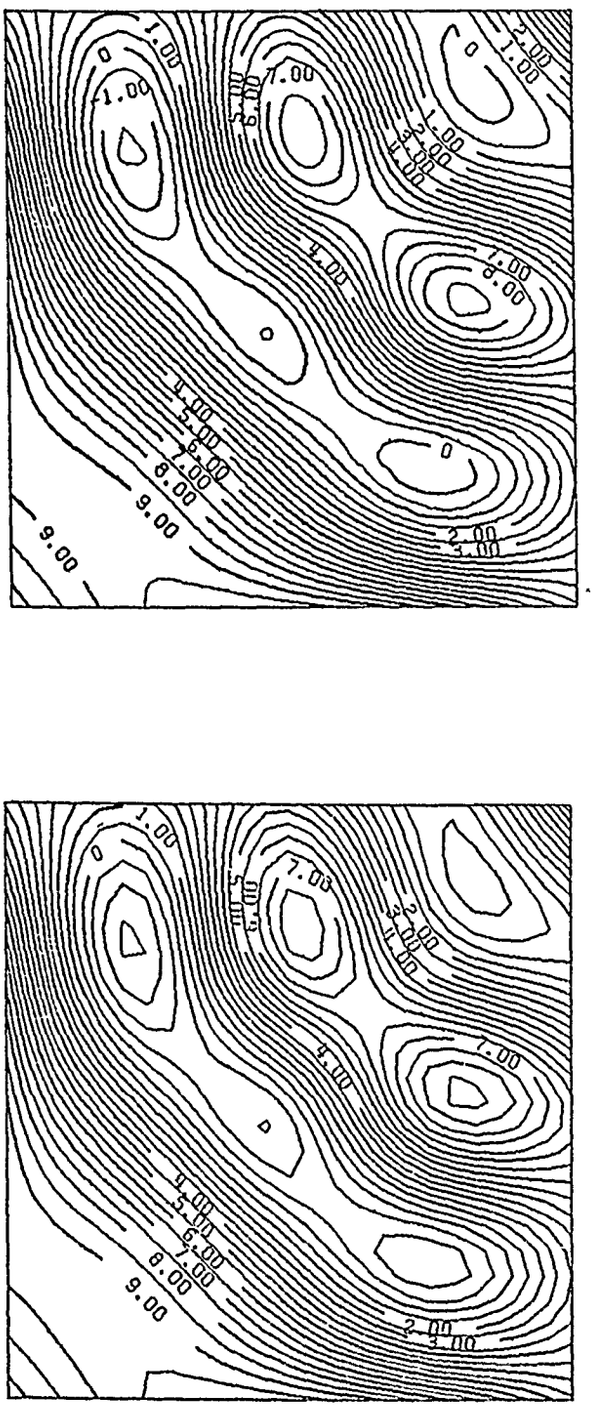


Figure 2.8.1 Smoothed and non-smoothed contours

2.8.2 Contour mapping. Usually, the two-dimensional coordinates x and y are interpreted as cartesian coordinates and the mapping "data space → image space" (contour plot) is assumed to be given by

$$x^* = c \cdot x$$
$$y^* = c \cdot y \qquad (2.8\text{-}1)$$

where (x, y) are the coordinates of a data point, (x^*, y^*) are the coordinates of the plotted point, and c is the constant scale factor.

However, especially in all earth sciences the coordinates are not to be understood as cartesian coordinates but as, e.g., spherical coordinates φ, λ, and there is the wish or need to choose a mapping different from that in equation (2.8-1). For example, a Mercator projection

$$x^* = c_1 \ln \left[\tan \left(\frac{\pi}{4} + \frac{|\varphi|}{2}\right)\right] \cdot \operatorname{sign}(\varphi)$$

$$y^* = c_2 (\lambda - \lambda_0)$$

with

φ, λ = spherical coordinates (latitude, longitude),

λ_0 = longitude origin

c_1, c_2 = scale factors in x and y- direction;

Another example, a linear transformation of the form

$$x^* = c_1 (x \cos \alpha + y \sin \alpha)$$
$$y^* = c_2 (-x \sin \alpha + y \cos \alpha)$$

with the constant azimuth α as rotation angle. Obviously, there is an unlimited variety of possibilities for coordinate transformations (mappings). If any mapping different from (2.8-1) is desired, the user has to define this mapping by providing the corresponding subroutine and he has to inform GSPP that a mapping is requested. This mapping has, naturally, to be done for all points of the contour; actually, it is performed after the clean-up of the contour point vector. Fig. 2.8.2 shows a Mercator projection of the contours in Fig. 2.8.1.

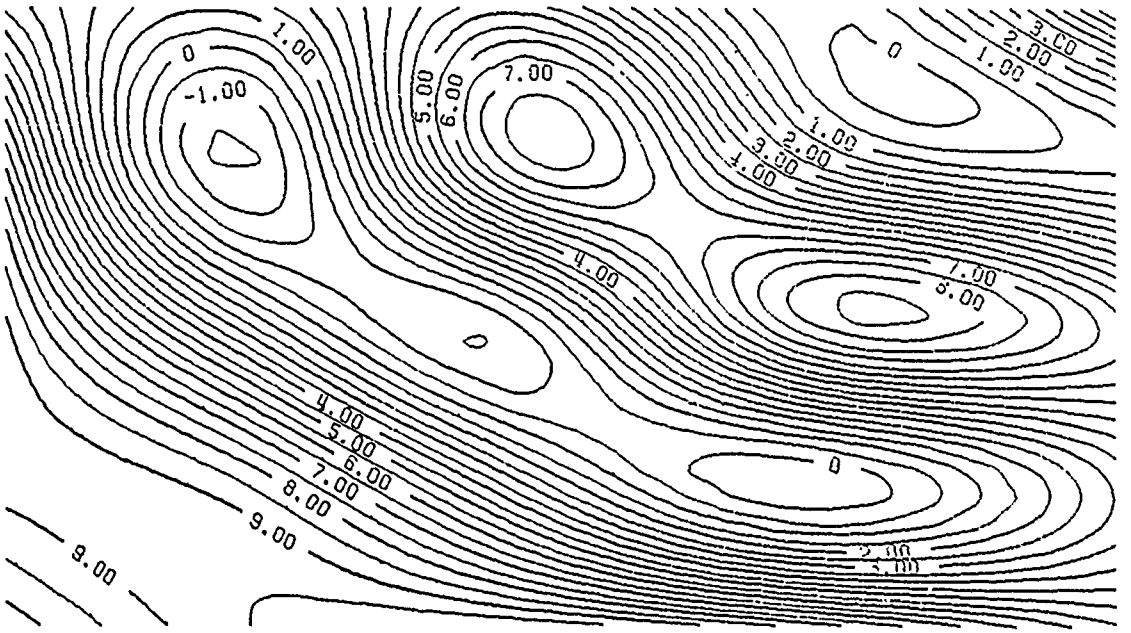


Fig. 2.8.2 Mercator projection of the contours shown in Fig. 2.8.1

2.8.3 Contour labeling. By labeling of contours we understand the plot of the contour value into an interval which is kept free of the contour line. The contour has to have a sufficient length such that a contour value fits in an interval which is smaller than the actual length of the contour. If a contour is too short, no label (contour value) will appear. The direction of the label is designed to be identical with the tangent to the contour at the midpoint of the label.

The length of the label depends on the actual contour value, the number of decimal places to be plotted, and the sign of the contour value, according to the equation

$$w = c \cdot h \cdot \begin{cases} n+2+u(-x) & \text{for} \quad |x| \leq 10.0 \\ n+2+\text{int}(\log_{10}(|x|)) + u(-x) & \text{for} \quad |x| > 10.0 \end{cases} \qquad (2.8\text{-}2)$$

with

w = label length,

h = symbol height,

c = ratio symbol length/symbol height

x = contour value,

n = number of decimal places to appear,

int = integer function

$u(x)$ = unit step function, $u(x) = \begin{cases} 1 \text{ for } x > 0 \\ 0 \text{ for } x \leq 0 \end{cases}$.

If not defined explicitly by the user, the number of decimal places is chosen such that at least 3 significant digits appear. If the absolute contour value is greater than or equal to 1000, no decimal digit and no decimal point will appear. If the contour value is zero, only a "0" without decimal point and decimal digit will be plotted (see Fig. 2.8.1).

Moreover, labeled contours can be plotted with multiple linewidth (if the plotter is designed for such a purpose).

The labeled contour interval has to be an integer multiple of the non-labeled contour interval ; the default value in GSPP is each second contour labeled with double linewidth.

2.8.4 Contours within a window. Usually, the complete bicubic spline surface from m=1, ..., M and n=1, ..., N will be contoured. Sometimes, however, one is probably interested in only a part of it in order to see more details when plotted in a larger scale or one likes to have more contours than usual plotted in a particular area. For such purposes GSPP offers the possibility to plot only a rectangular

part of the whole surface, say the window $m = m_0, \ldots, m_1$ and $n = n_0, \ldots, n_1$. Figs. 2.8.3 give an example of such a window plot.

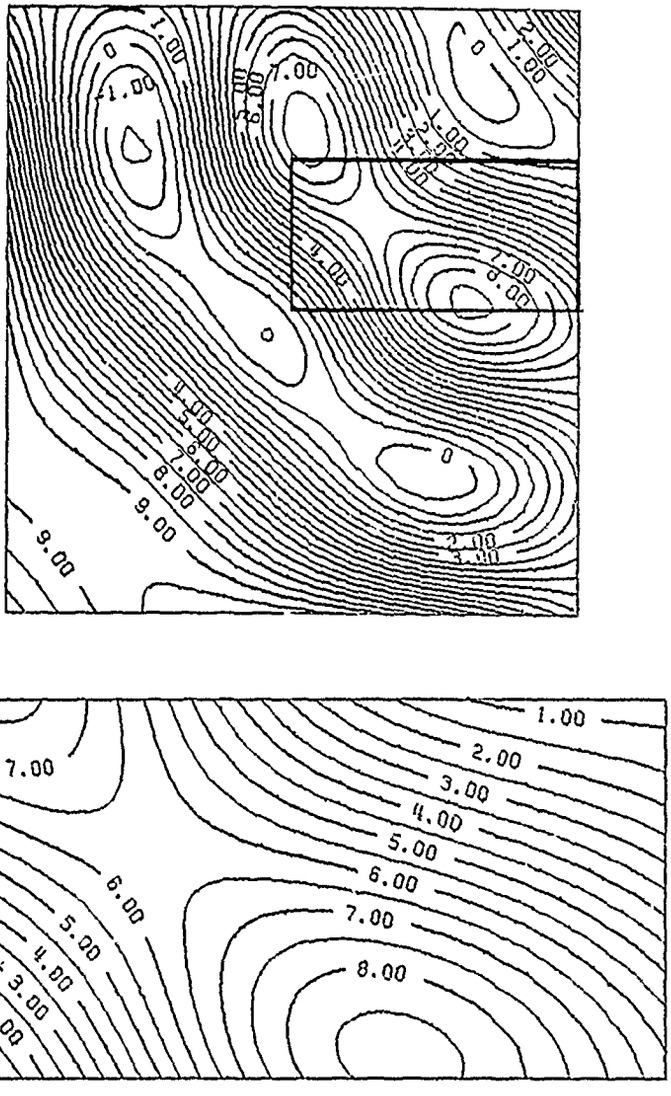


Figure 2.8.3: Contours within a grid window .

This feature, however, is limited in many respects: it allows only rectangular windows on full grid elements (from one grid point to another grid point). Furthermore, it is not possible to suppress the contour drawing within a rectangular region. In order to overcome all these deficiencies, a highly sophisticated routine has been developed and incorporated into GSPP; it will be described in the following section.

2.8.5 Contour-free regions of arbitrary shape. Sometimes there is the desire to plot the contours only within a predefined closed region, or vice versa, to suppress the contour drawing within certain regions. For example, it might be requested to plot a gravity anomaly map only for the state of Ohio or, a rather recent application, to generate a world geoid solely based on altimeter data and to plot this geoid only over the oceans and greater lakes because of the lack of data over the continents (e.g., Rapp, 1979). Before we describe the procedure which is capable of doing this, let us mention that such a statement is very simple to make -- its translation into a computer language, however, is extraordinarily complicated. The reason is primarily that such a region is, in general, not convex, practically arbitrary in shape, and last but by no means least, may consist of a very huge number of boundary points. In order to give an idea: the detailed world shore-outline data bank consists of some 80 000 points. Moreover, there might be contour-regions within contour-free regions, and so on -- this could be the case in the above mentioned altimeter example: geoid plot only over "water": ocean (continent (lake (island (lake (island ...))))). More technically, the procedure should be able to handle sequences of subsets

$$A_0 \supset A_1 \supset A_2 \supset \ldots \supset A_n \qquad (2.8\text{-}3)$$

such that contours are drawn only on

$$A_{2k} - A_{2k-1} \qquad k = 0, 1, \ldots, \frac{n}{2} \qquad (2.8\text{-}4a)$$

or only on its complement

$$A_{2k-1} - A_{2k} \quad , \quad k = 1, 2, \ldots, \frac{n}{2} \tag{2.8-4b}$$

(see Fig. 2.8.4). Sometimes it is even required that the contours are drawn (or suppressed) on a union of mutually exclusive (disjoint) sequences of subsets. In this case A_k and n depend on a parameter i (see Fig. 2.8.4):

$$A_0^i \supset A_1^i \supset \ldots \supset A_{n_i}^i \quad , \quad i = 1, \ldots, I. \tag{2.8-5}$$

GSPP is capable of handling even this case.

Honestly speaking, the author himself had absolutely no reasonable idea how to start when he got involved in this problem. It is clear that, in order to plot contours only in certain regions, it is necessary to a) find the intersection between the contours, and b) to know which part of the contour has to be plotted and which one has to be deleted. In order to find the intersections between a contour and all region boundaries it is necessary to develop a stable algorithm which is capable of finding the intersection between two straight lines, and to run this algorithm for all possible line segments of the contour and all possible line segments of the boundaries. If the number of boundary points is very small, this procedure will not be very expensive; however, if the boundaries consist of 100 points and more, this method becomes absolutely prohibitive in terms of computer time. How else should one attack this problem? Since in nature almost everything is optimized (very probably because of the long time of evolution), it came to the author's mind to analyze the almost automatic human decision process for the particular problem of suppressing the drawing of contours within predefined regions, and to "translate" this process step by step in the computer language FORTRAN IV. The analyzation of the decision process was relatively simple, not so the translation into FORTRAN IV.

Let us first analyze the process of contour suppressing in terms of human action, the interplay between the memory (brain memory), the optical sensor (eye), and the mechanical part of the pen movement (drawing by hand): First of all we obtain from an external unit (somebody) the surface information, generally in terms of function values at grid points, together with the boundary outlines and the information in which region the contours should be drawn (or in which region they should be deleted). This corresponds to the information represented in Fig. 2.8.4 (+ surface information).

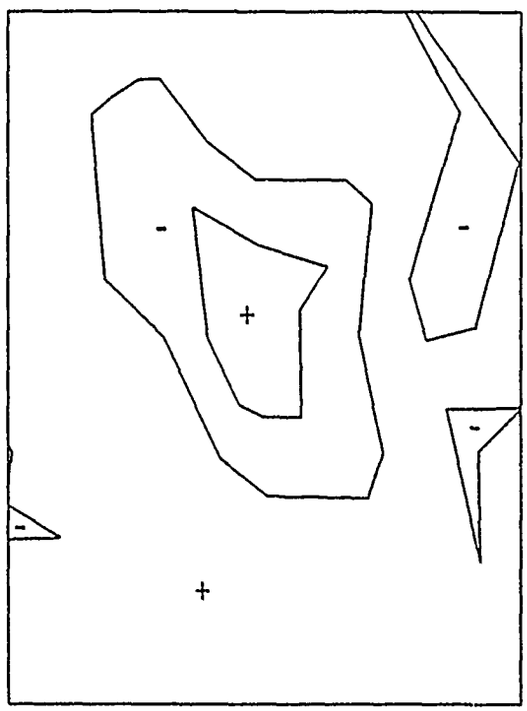


Figure 2.8.4 Boundary outlines

+ ... contour region

- ... contour-free region

1) The first step consists of a global "look" at the situation in order to get an impression of where contours have to be drawn. If no other means is available, the test-person will assign a "YES" to all contour regions, a "NO" or just nothing to all contour-free regions, and a "?" to the boundary zone which is neither YES nor NO, and "store" this YES-?-NO pattern in his brain. Now we have the following information available: the array of function values at the grid points ($\hat{=}$ the surface), the rough pattern of YES-?-NO entries, and the exact boundary outlines.
2) The next step is the decision where we should start searching for contours. The YES-?-NO pattern is the guideline for this decision. Obviously, one does not start amidst a NO- group in order to find out, after the calculation of the contour line element, that it was useless anyway because of its location within the contour-free region. The test-person would probably also not start to calculate and draw contours amidst a YES- group -- this would speak against a systematic solution of the problem. It would start rowwise (or columnwise) with a "?" element, an element from a boundary zone. In other words, the test-person starts to calculate contours in a "zone" along the boundaries.
3) He calculates the line element of the contour and has now to intersect this line element with the boundary (or the boundaries). For this purpose he needs to have the "accurate" information about the boundary (boundaries) in this small region. He does not need to know the whole boundary information, but only a "close-up" of a very limited part.
4) He moves his eyes closer to this part, concentrates on only two lines, the contour segment and the small part of the

boundary, and forgets for some time all other regional and global information. He finds the intersection; now he has to decide on the pen position, up or down. Since he "forgot" in the meantime all global information, he recalls them again by moving his eyes away from the picture in order to get again the global impression of the YES-?-NO pattern. This pattern, in turn, enables him to choose the correct pen position, and secondly, to make him trace the contour in the correct direction (the direction of pen down).

5) He lowers the pen at the point of intersection, calculates the next contour element; if this happens to be in the "?" region he repeats steps (3) and (4); if it falls in the YES - region he continues drawing until he finds the next "?" region. The first pen-up/pen-down decision is sufficient for all further ones because of the alternating character of the contour/ contour-free region sequence.

These 5 steps described above are in principle "translated" into FORTRAN IV. In the sequel we present the essence of this translation.

1) As described in Chapter (2.7), an integer * 2 array is assigned to the array of bilinear elements. The same array is used for storing the information whether a particular element (here rectangle on which a bilinear function is defined) happens to be located completely inside a contour region (YES), completely inside a contour-free region (NO), or if the element is crossed by one of the region boundaries.

The boundary information consists of the number of boundaries, the information whether boundary i encloses a contour - or contour free region (for all boundaries i), the boundary coordinates and information if the boundary coordinate sequence runs clockwise or counter-clockwise.

In the sequel all elements are marked by a negative number (corresponding to the question mark "?" above), if at least one boundary line crosses the element. The absolute value of this negative number serves at the same time as counter of how many boundary points have been found (or interpolated) within the element in consideration. The interpolation of boundary points is necessary if their mutual distance is, loosely speaking, too large relative to the size of the element; the interpolation is linear, for reasons of simplicity. As soon as the first boundary has been completed (all boundaries need to be closed) and the "border zone" has been marked, the region is filled up with "1" 's ($\cong$ NO) if the region is to be contour-free; the contour-regions bear a "0". After all regions have been processed, a pattern of "?", "0", "1" is obtained. This pattern represents the global information of where contours have to be drawn and of where they should be deleted. Fig. 2.8.5 shows such a pattern associated with the boundary situation of Fig. 2.8.4.

In addition, all boundary points (primary and interpolated) which will be used to calculate contour-boundary intersections are stored on an auxiliary vector. (Usually, not all boundary points are needed, unless the boundaries are completely inside the total rectangular plotting range; e.g. consider the case of the complete world shore-outline data bank and a plot restricted to Europe; then probably only a couple of thousand out of the total 80 000 boundary data will be used.)

2) The next step is already a part of the actual contouring. The array filled up with negative numbers (boundary zones), zeros (contour regions) and ones (contour-free regions) is

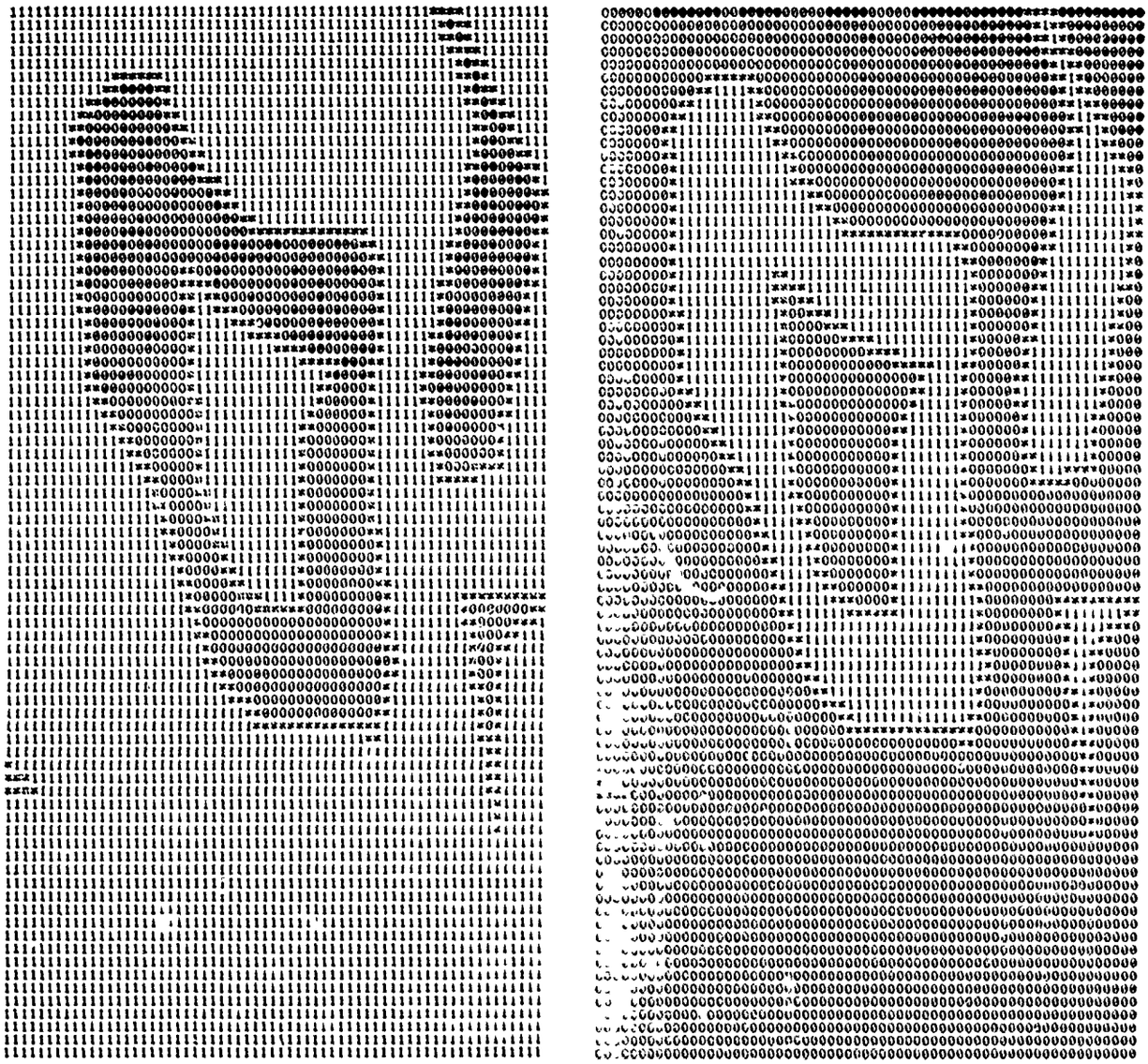


Figure 2.8.5 Region patterns

checked, prior to the function value-contour value comparison; all elements filled up with "1" are branched; all contour points are calculated and stored for successive "0" elements.

3) As soon as the contour touches a negative valued element (boundary zone), the program switches to a "close-up" look for this particular area. The close-up region covers only all boundary data within the element in consideration plus all boundary data in the immediate neighborhood of this element; This number of data is always very small indeed (recall that one element has a maximum size of 3x3 mm). At this point we do not go into detail, this would be a little bit too technical. The whole close-up look consists in a series of logical operations involving a tree of pointer vectors; the only calculations are integer additions -- therefore, this procedure is extremely efficient and takes very little computer time.

4) Having assembled all local boundary points, the program continues with the actual calculation of intersections which is performed in a stable subroutine. Such a line intersection subroutine was available to the author in form of an elegant flowchart (Neubauer, 1978). The subroutine following this flowchart was programmed. If no intersection has been found, the program continues and branches to the next element. If one intersection has been found, there is the problem of deciding whether at this particular point the pen has to be raised or lowered. This decision is enabled by a global information in terms of a logical vector, assigned to the boundaries, which provides information about the region behavior in a direction orthogonal to the boundary (contour or contour-free); e.g., in Fig. 2.8.6 the points P_1

P_2 are contour points, #37, ..., #42 are boundary points of the boundary B_3; they represent, for this particular element, the "close-up" ; (37, ..., 42) are only a part of the boundary B_3. The global information consists in the knowledge of which side of the boundary B_3 is contour-free; then it is clear that at S the pen has to be lowered and the connection $S\text{-}P_2$ has to be plotted. The part $P_1\text{-}S$ of the contour will be deleted.

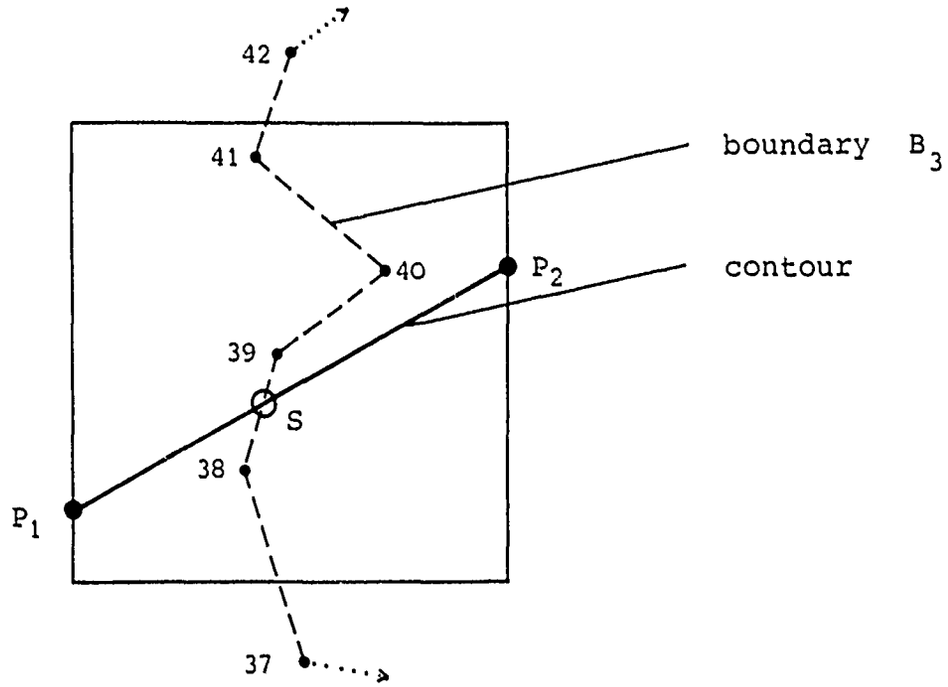


Fig. 2.8.6 Close-up contour/simple boundary

The case of a single intersection point per element is relatively easy to handle. Very detailed and rough boundaries, however, may cause more than one intersection points. Moreover, the intersections may even refer to different boundaries. In such a case the close-up, the retrieval of the logical information associated with each boundary, the calculation of all intersection points and the pen position logics are much more complicated.

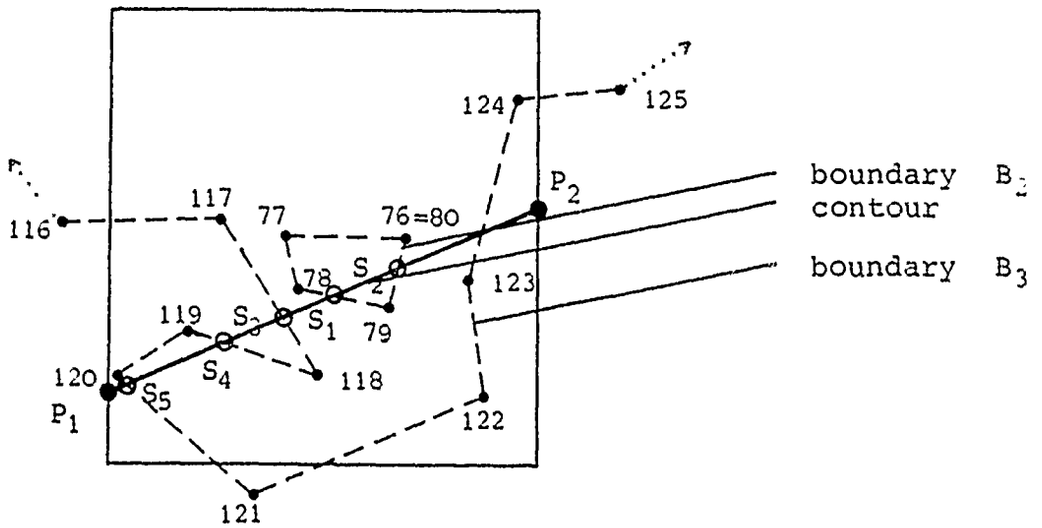


Fig. 2.8.7 Close-up contour/complicated boundary

E.g.: In Figure 2.8.7 the points P_1 and P_2 are contour points as before. But now there are two boundaries involved, B_2 and B_3. The close-up of B_2 is identical with B_2 -- it is a closed boundary within a single element consisting of the points # 76, ..., #80 (the points #76 and # 80 are identical because the boundary is closed). The close-up of B_3, however, shows only a part of B_3 -- the points #116, ..., #125. Altogether, 13 boundary line elements are to be checked for an intersection with the contour; 6 intersection points S_1 ..., S_6 have been found. Let us assume that boundary B_2 encloses a contour region; clearly, the region to the right hand side of boundary B_3 has to be also a contour region because of the alternating

behavior of such regions. Let us now intersect all boundary line elements with the line $\overline{P_1 P_2}$, starting with boundary B_2 in increasing order. Then it can be seen immediately that the sequence of intersection points $\{S_i\}$, i=1, ..., 6 is more or less arbitrarily distributed between $\overline{P_1 P_2}$ (S_5, S_4, S_3, S_1, S_2, S_6). In order to make the pen position decision it is necessary to know just one intersection point together with the boundary logics; all other pen positions follow from the alternating behavior. This alternating continuation, however, is only possible if the actual intersection point sequence is known. This can be accomplished by first transforming the intersection point coordinates into a new coordinate system whose origin is, in principle, arbitrary and whose x-axis is parallel to the line $\overline{P_1 P_2}$, and then re-ordering the intersection points according to increasing x-coordinates. The first operation is just a rotation of the original coordinate system, the second operation is not so trivial --it involves a sorting algorithm. GSPP uses an algorithm written by P. Meissl which has been adapted for this particular problem. In the case discussed above and graphically shown in Fig. 2.8.7, the intersection point S_5 belonging to the boundary B_3 would have the smallest transformed x-coordinate among all other intersection points. This point will also be taken for the pen-position decision process: From the global information it is known that the region to the right hand side of the boundary B_3 is a contour region; consequently, the segment $\overline{P_1 S_5}$ has to be plotted, $\overline{S_5 S_2}$ deleted, $\overline{S_4 S_3}$ plotted, ... and so on. This completes step (4),

5) In the sequel the next element will be checked. If it has a value "0" assigned, the contouring can continue without the need of close-up's and intersection calculations; the same

is true for values "1" -- in this case the program can "jump" from one element to the next until it finds all elements with a negative value in which case steps (3) and (4) have to be repeated.

If smooth contours are requested, the program has to "remember" all intersection points including at least the pen position at 1 intersection point. The parts of the contour which have to be plotted are then smoothed along the procedure described in section 2.8.1. The following Figure 2.8.8 serves as an illustration. Figure 2.8.9 shows that even for regions with very complicated boundaries the procedure works absolutely correct and gives total resolution.

2.8.6 Region boundary plot. Besides the suppressing of the contour plot within certain regions, there is also a routine built in which plots the actual boundaries. There is an option to plot some of them; there is the option to assign to each boundary a certain linewidth. In any case the boundary lines will be clipped off at the border of the rectangular plotting domain. The boundary points will be connected linearly throughout.

The clipping process has been designed in the following manner: assume the rectangular plotting domain R to be described by the coordinates $[x_0, x_1, y_0, y_1]$. If a boundary line happens to be completely inside the rectangle R, it will be plotted (after an optional mapping) immediately. If a boundary line happens to be completely outside the rectangle R (all x-coordinates $< x_0$ or all x-coordinates $> x_1$ and/or all y-coordinates $< y_0$ or all y-coordinates $> y_1$), the program skips this boundary and continues with the next one. If none of the above conditions is true, the boundary either crosses the border of the rectangle R or it is still completely outside R. The same is true for all line segments of the boundary.

The following number pattern is assigned to the nine rectangular regions (1 closed, 8 open) generated by the boundary lines of the plotting rectangle and its infinite continuations:

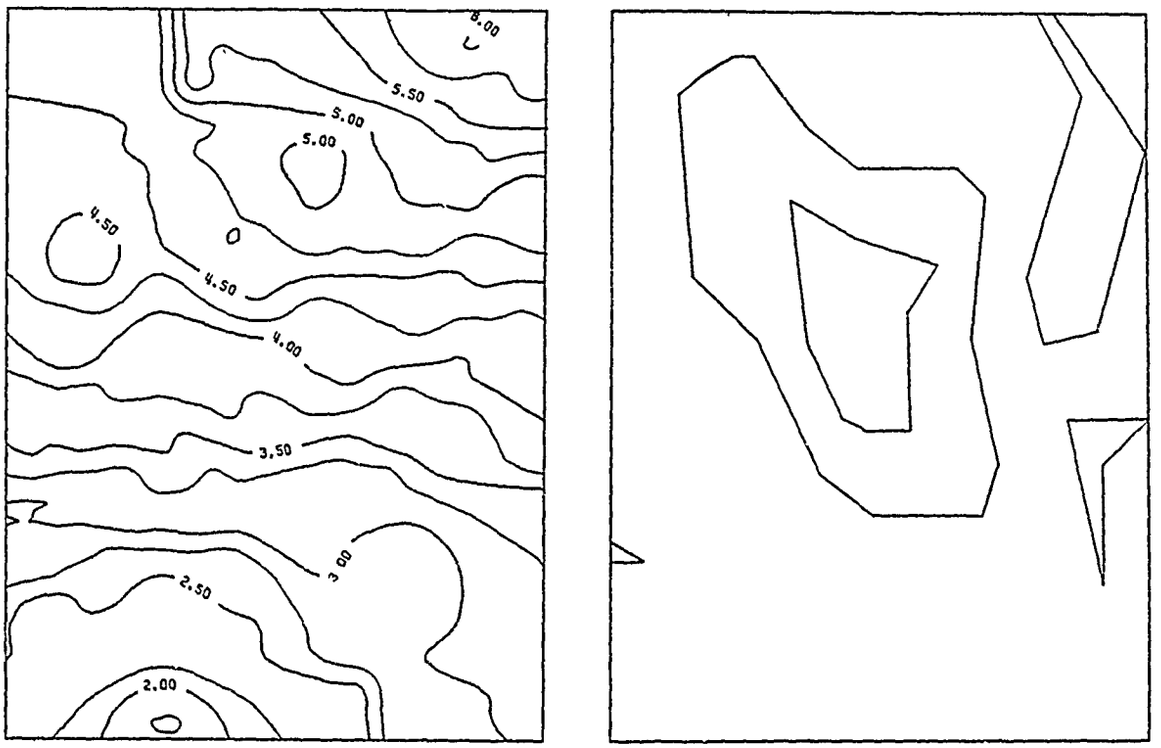
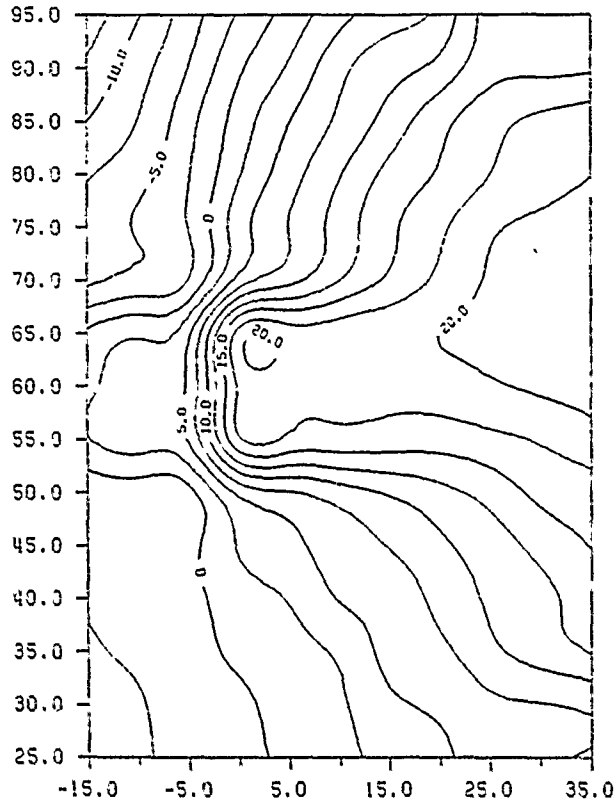
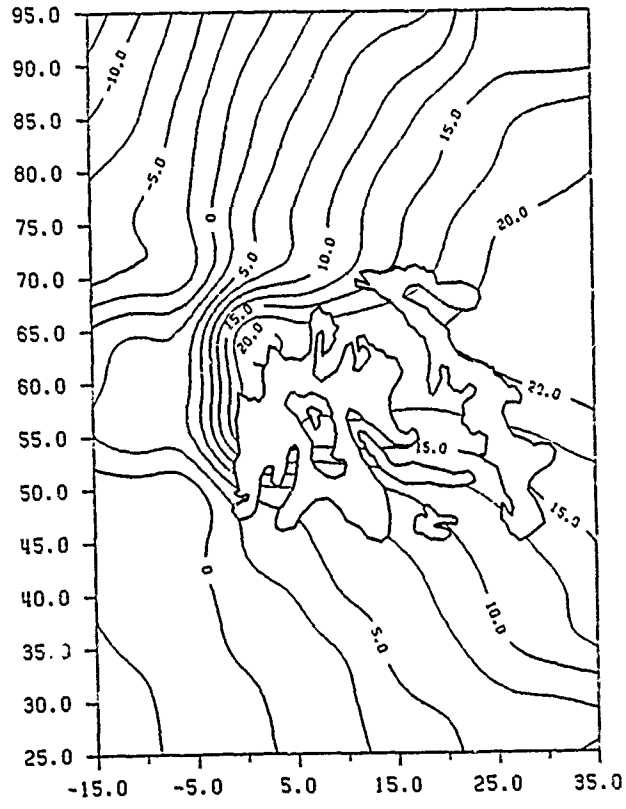


Figure 2.8.8: Contours, boundaries, contour suppressing (simple boundaries)





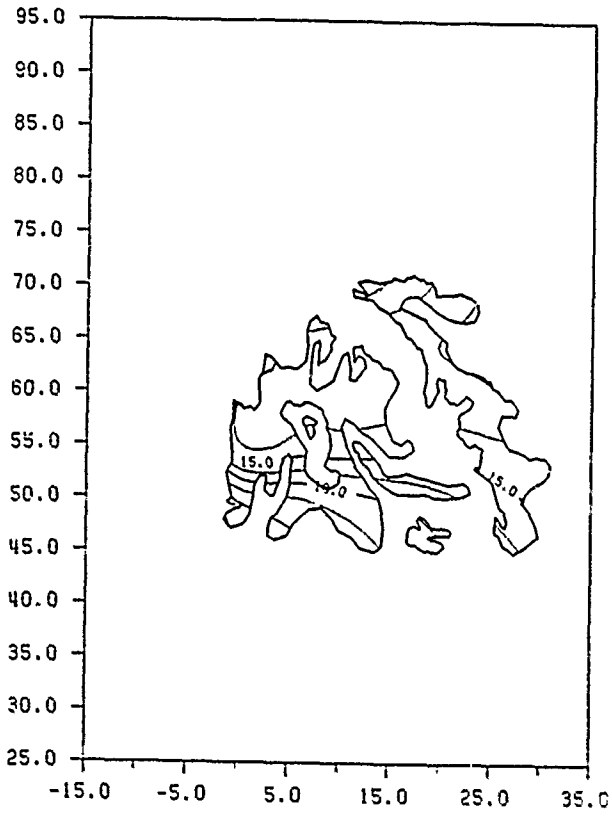
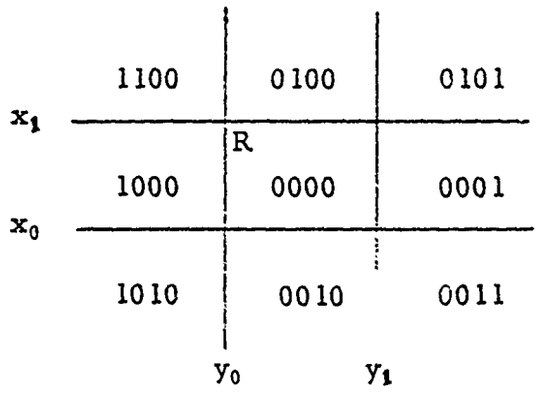


Figure 2.8.9: Contours and complicated region boundaries



One of these values is assigned to the boundary points according to its position. Then it can be immediately verified that a line segment is definitely completely outside R if the number, which results from an addition of the codes of the two line segment points, contains at least one "2"; if it does not, it may be completely outside but might also be partly inside R, and therefore, intersect the boundary rectangle. If the sum is zero, it is completely inside R. E.g.:

> P_1 is in the region 1010, P_2 in 0011; the code sum is 1021 -- therefore, the line $\overline{P_1P_2}$ is completely outside R. If P_2 is in 0101, then the code sum is 1111 -- therefore, the line $\overline{P_1P_2}$ could cross the boundary of R.

If the line segment code contains a "2", the line segment will be deleted; if it contains only zeros, it will be plotted; if it contains at least one "1" but no "2", the program finds the point(s) of intersection with the help of a line intersection algorithm described in (Neubauer, 1978). If a mapping is defined and requested, all boundary points (and intersection points) will be mapped; their connection will always be linear regardless of the mapping.

2.8.7 Plot of horizontal and vertical axes. The axis routine will be described in detail in Chapter 5 of Part B. Here we mention that a couple of axis options are available for a contour plot.

Any axis plot can be suppressed; the axis can be obtained on the left hand side and on the bottom of the contour plot; the axis can, in addition, be obtained on the right hand side and on the top of the contour plot. The axis itself consists of a straight line (which can also be suppressed), tick marks and scale numbers. The tick mark interval, the tick mark length and the tick mark direction can, within certain bounds, be chosen by the user. If the user does not define a tick mark interval, it is taken as the grid interval (for the surface representation). The scale numbers can be plotted in 4 different directions ($k * \pi/2$, $k = 0, \ldots, 3$); the symbol heights and the number of decimal places can be defined by the user, otherwise default values will be assigned (e.g., such that at least 3 significant digits are plotted). A scale number will always be centered with respect to its corresponding tick mark, where the actual length of a scale number is determined by equation (2.8.2). If the sequence of scale numbers is too dense (such that overlapping would occur), certain scale numbers will be deleted. In addition can the distance between the tick marks and scale numbers be chosen by the user (within certain bounds). Non-labeled tick marks will be 40% shorter than labeled ones. No axis will be plotted if a contour mapping is involved.

2.8.8 Plot of a grid superimposed on the contour plot. In order to make graphical interpolations of function values between the plotted contours easier, a grid plotting routine has been included in GSPP. The following grid patterns are available (Fig. 2.8.10): full one, dashed line with arbitrary dash length and interval between dashes, and a plot of open crosses at the grid points. The grid distance is always identical with the tick mark interval. If a grid plot has been requested and no grid parameters have been defined, a full line grid will be plotted. No grid will be plotted if a contour mapping is involved. Further details about a grid plot can be found in Chapter 6 of Part B.

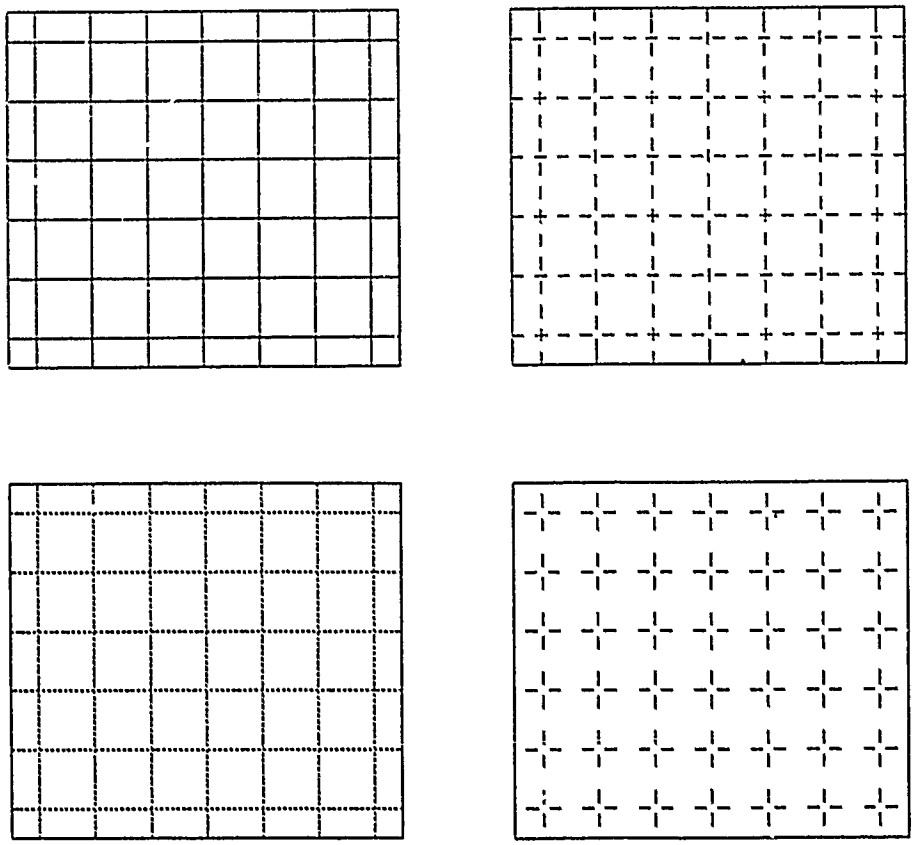


Figure 2.8.10: Grid patterns

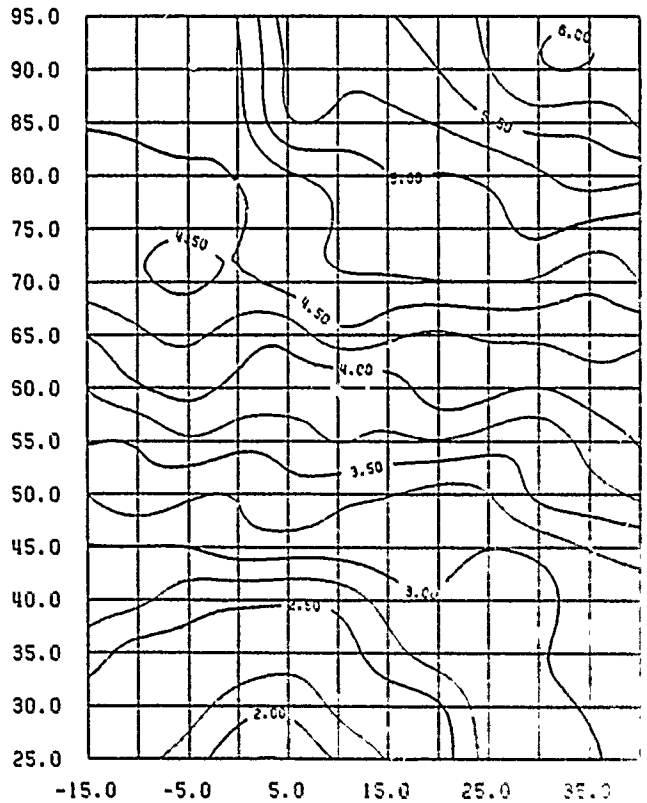


Figure 2.8.11: Contours superimposed by a grid .

2.8.9 Plot of data superimposed on the contour plot.

Especially when a surface is derived from a set of homogeneous data (noisy or not), it is often of interest to plot the data with the contours as background; this enables the user to check the goodness of surface fit relative to the data. But also a plot of the data point locations and contours might be quite useful if one wants to see how the prediction works in data-free areas or in areas with poor data coverage. In order to satisfy all needs, GSPP offers the following 3 options: only data position symbol plot, data position symbol plot and data number plot, data position symbol plot and data number and data value plot.

The position symbol can be chosen among all symbols available (differs from one plotting software package to the other) -- some of them (0, ..., 13 for IBM software) give centered symbols, some of them don't. The data number and the data value will always be symmetrically located relative to the data position, the data number above the data position, the data value below the data position. The symbol height can, within a certain bound, be selected by the user; the same is true for the number of decimal digits for a data value plot. In any case, the symbol height of the data number will be half the symbol height of the data value; this is to distinguish them more easily.

Besides the data plot there is also the possibility to plot the predicted data in the same way as described above (plot the grid point locations and predicted value at the grid points).

If a mapping is involved, the data coordinates will be mapped in the same way as the contour point coordinates. In any case a data point outside the rectangular plotting domain will never be plotted. Figure 2.8.12 illustrates a data plot superimposed on a contour plot.

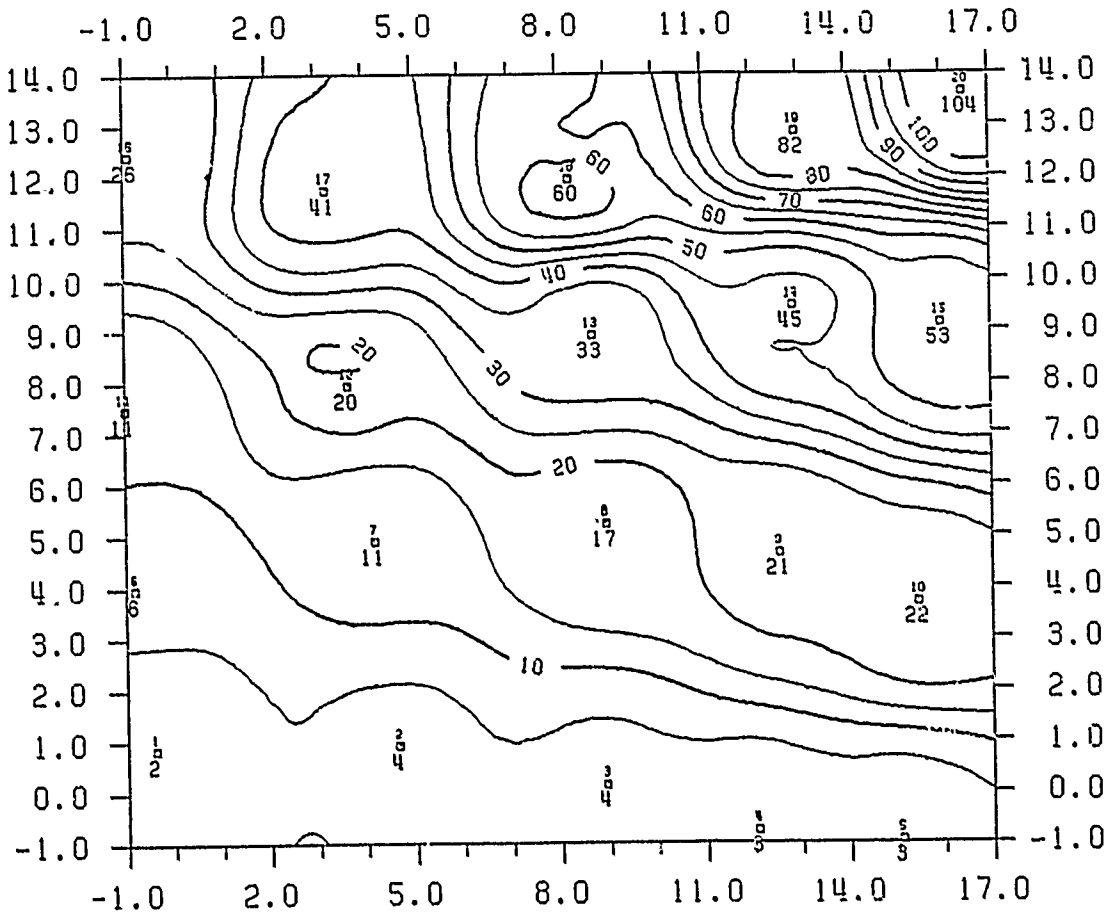


Figure 2.8.12 Contour and data plot

2.8.10 Title and label control and plot. Titles and labels are used to identify plots. For this reason GSPP offers the possibility of a title plot, consisting of up to 10 title lines, as well as single line

horizontal and vertical label plots (a line is to be understood as a maximum of 80 characters = information stored on one punched card). The height and line width of the symbols can be chosen by the user within certain bounds.

The title plot routine itself is intelligent. It offers the title to be plotted in the following four ways: same as on the input cards (default), left justified, centered, right justified regardless of how the title information was actually punched. If the title length exceeds the actual length of the contour plot, the symbol size will be reduced. If the reduced symbol height happens to be smaller than the allowed lower limit, the lower limit will be chosen and the title will start on the left side regardless of the requested mode. The symbol size will also be reduced automatically when the total title height exceeds upper limits, fixed by GSPP. The title will always be plotted above the contour map; its position, relative to the upper left corner point of the rectangular plotting domain, can be chosen by the user -- if not, default values will be assigned.

Two single line labels, one along the horizontal, one along the vertical axis can also be plotted. As long as the label length does not exceed the corresponding length of the contour plot, the label will be centered relative to the contour plot. If it exceeds this length, the symbol height will be reduced automatically, but not below a minimum height defined in GSPP. If the label length, with minimum symbol height, is still bigger than the corresponding length of the contour plot, it will no longer be centered; instead, it will be left-justified.

Usually, the labels will be plotted along the left vertical and along the lower horizontal axis; however, when axes are to be plotted all around, also labels will be plotted all around. In this case the title will also be shifted automatically in order to avoid a possible overlapping. No title and no labels will be plotted if a contour mapping is requested.

Further information about the title routine can be found in Chapter 7 of Part B.

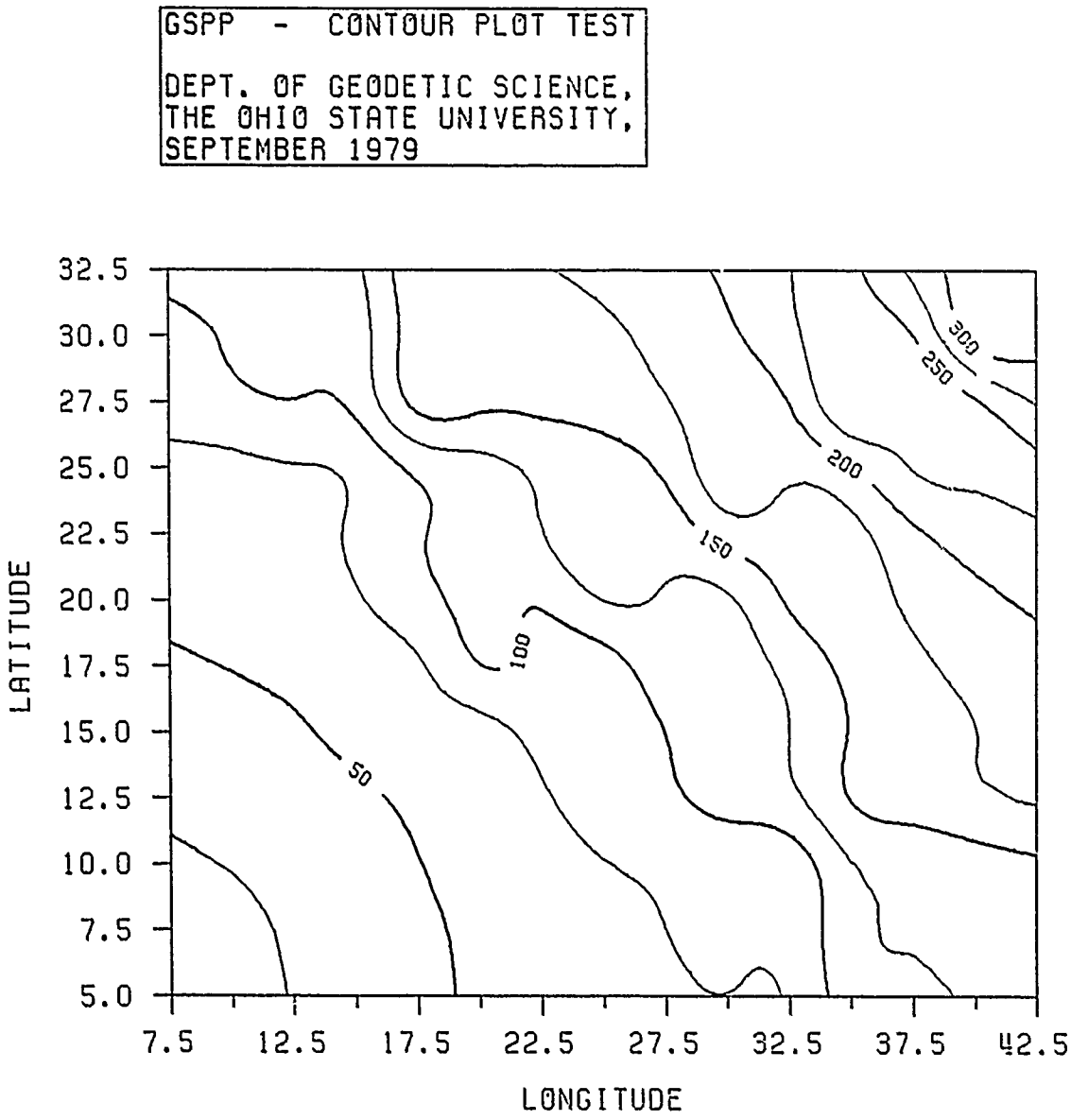


Figure 2.8.13: Contour, title, and label plot

2.8.11 Contours of surface derivatives. The contouring part of GSPP has been designed primarily for a contour plot of the surface itself. The surface is defined as a bicubic spline function which is determined by function values at the grid points of a regular rectangular grid. For the contouring it is of no concern whether the grid values were known in advance or if they were predicted on the basis of other, probably irregularly distributed inhomogeneous data, or if they are function values derived from a least-squares regression polynomial. All of them represent a surface which is interpreted as a bicubic spline function.

Apart from surface contours, GSPP also offers contours of surface derivatives up to and including a second derivative in both coordinate directions. Therefore, contour plots of the following surface derivatives can be requested from GSPP:

$D_x f,\ D_y f,$

$D_{xx} f,\quad D_{xy} f,\quad D_{yy} f$

$D_{xxy} f,\quad D_{xyy} f,$

$D_{xxyy} f\ .$

The derivatives are calculated according to the set of equations (2.6-7b), which are derivatives of a bicubic polynomial. Since a bicubic polynomial is twice continuously differentiable with respect to both independent variables, the highest derivative offered, $D_{xxyy} f$, is still a continuous function; it is a continuous and piecewise bilinear function and therefore, a hyperbolic paraboloid (cf. Section 2.7). All lower order derivatives will be functions of higher order, and therefore, smoother.

The following Figures 2.8.14(a, b, c) show a contour plot of a surface, its first derivative in x-direction and its highest allowed derivative $D_{xxyy} f$. From the $D_{xxyy} f$ plot one can see very clearly the contour's tendency to run in directions along the asymptotic lines of

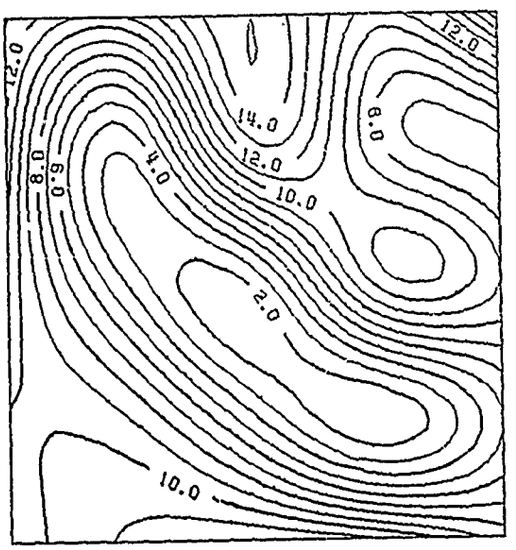


Figure 2.8.14a: Surface contours

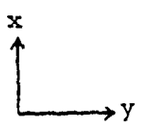
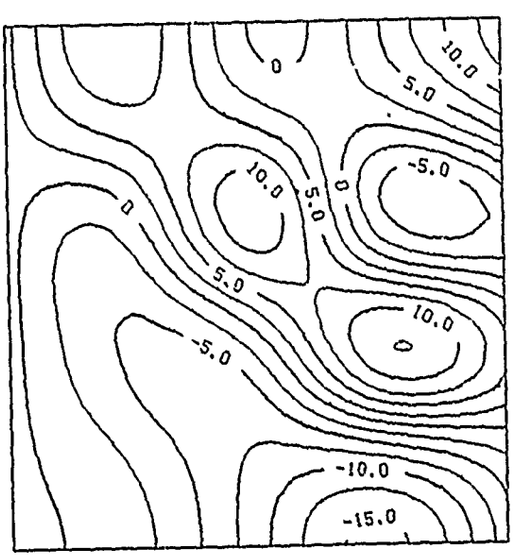


Figure 2.8.14b : Contours of D_x-derivatives of the surface shown in Fig. 2.8.14a .



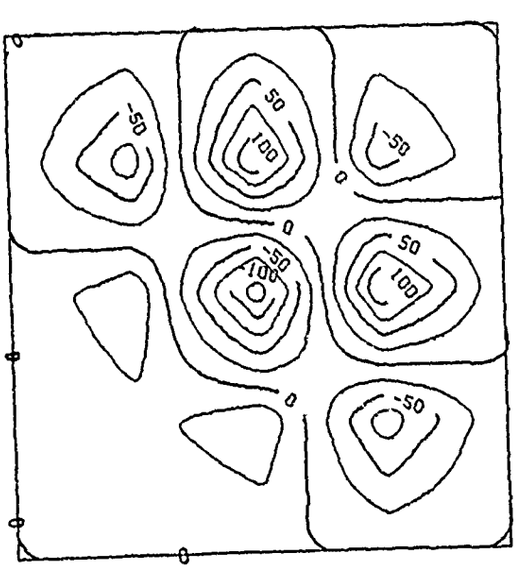


Fig. 2.8.14c: Contours of D_{xxyy}-derivatives of the surface shown in Fig. 2.8.14a .

the hyperbolas, which are parallel to the coordinate lines, and along the axes of the hyperbolas which span an angle of 45° and 135°, respectively, with the coordinate lines. These features are particularly pronounced in Fig. 2.8.14c because of the big size of the grid elements used (2x2 cm). This happens always when bilinear elements are contoured. Since the bicubic elements are approximated by bilinear elements, the same is true even in the contouring of surfaces. However, the size of the subgrid is kept so small that the effect cannot be seen anymore.

3. PROFILES

The term "profile" is usually thought of as a curve which results from an intersection between a surface and a vertical plane. Here we understand by profile a curve which can, but needs not necessarily be, a curve of intersection in the usual sense. If a curve is defined by a vector of pairs (x, y) with $y_i = y(x_i)$ function values at x_i, we speak about an "explicitly defined" profile. If a profile is defined by a surface (or data which are to represent a surface) together with profile start and end point, we speak about an "implicitly defined" profile. If more than one explicitly defined profile is to be plotted in the same frame, we will speak about a "multiple profile".

GSPP can handle these three types of profiles with a number of options and additional features like profiles of surface derivatives, profile derivatives, profile information, and many others, fully automatically. The following chapters describe these features in detail.

3.1 Explicitly defined profiles

As already stated in the foregoing introduction, an explicitly defined profile is thought of as a curve which is sampled by a vector of increasing arguments $\{x_i\}$, $i = 1, \ldots, I$ together with a vector of function values $\{y_i\}$, $i = 1, \ldots, I$. Of course, there is a number of ways to connect these sampled curve points. One could think of a polynomial interpolation or least-squares interpolation, etc. GSPP essentially offers three kinds: a simple sample point plot, a piecewise linear interpolation, and a cubic spline interpolation.

Profile derivatives can be requested up to the second order; derivatives will be derived from a cubic spline representation of the profile. The corresponding routines have been described in Sections 2.6.1 and 2.6.3. As far as the spectral content of the spline representation is concerned, the reader may consult Section 2.5.1.

The actual plot of the profile is performed within a window in x- and y-direction (argument window and function value window). If a window has been defined by the user, the curve will be clipped off when it leaves the window. The clipping algorithm is essentially identical with the one described in Section 2.8.6. If no window has been defined, the minima and maxima of the $\{x_i\}$ and $\{y_i\}$ vectors are assumed to coincide with the bounds of the window.

3.1.1 Optional profile procedures. The profile interpolation can be done piecewise linearly or by an interpolating cubic spline. For reasons of stability, the spline is calculated piecewise and overlapping if the number of data points is too large (see Section 2.8.1).

If no interpolation is requested, the profile points will be marked by a centered symbol which the user can choose.

The plot of horizontal and vertical axes is identical to that one described in Section 2.8.7 with the exception that no horizontal axis can be plotted above the profile plot.

Analogous to the superimposed grid in the contour plot, small crosses can be plotted at the intersections of grid lines (horizontal and vertical lines at tick mark intervals). These crosses will only be plotted below the profile. In addition, there is the option to draw a horizontal "zero-line" whenever the plotting window for function values contains the zero point.

Title and label plots are also identical to those described in Section 2.8.10.

As far as the plot of profile derivatives is concerned, there is the option to plot the first or the second derivative. Since the derivatives are taken from a cubic spline representation, the second derivative will still be continuous -- it is a piecewise linear function. The following Figures 3.1.1 (a, b, c) serve as illustration examples.

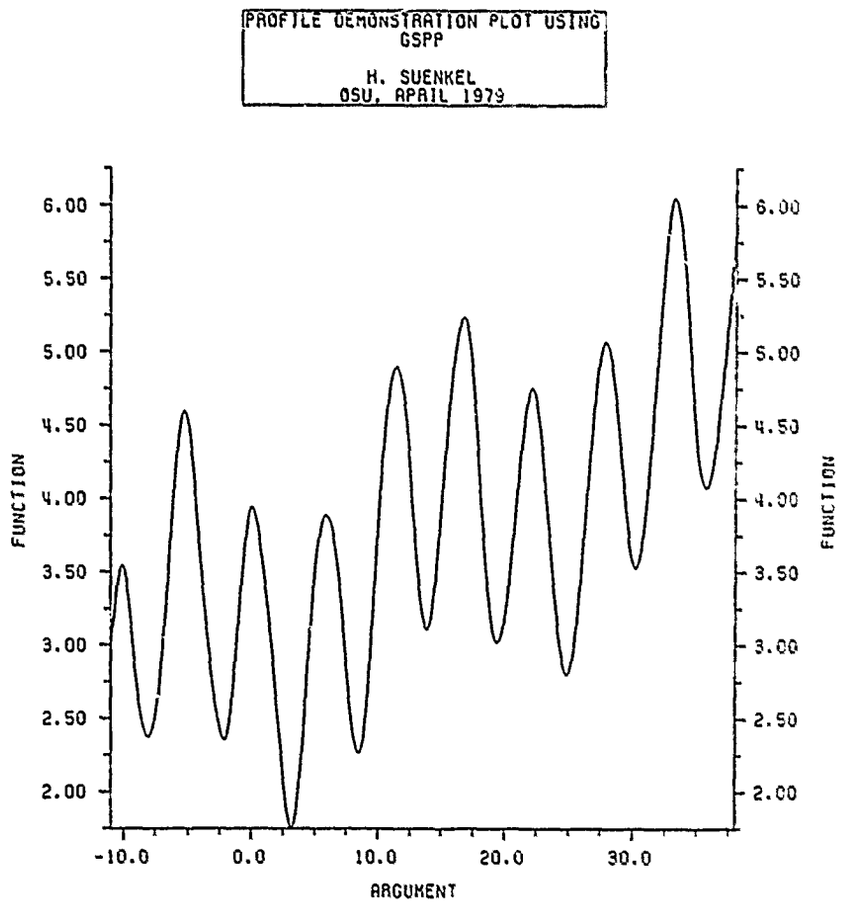


Figure 3.1.1a: Automatic profile plot plus title and labels; profile smoothing

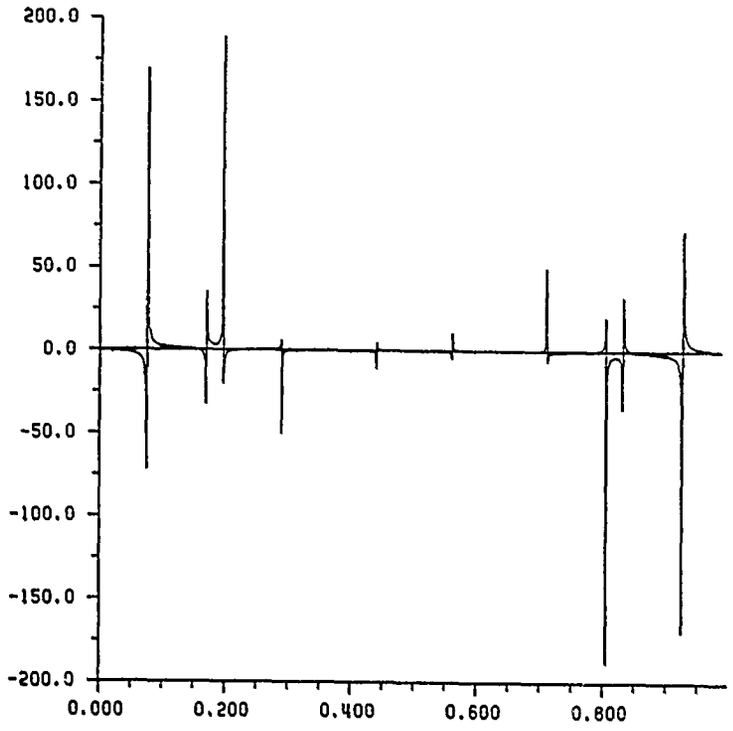


Figure 3. 1. 1b: Profile without smoothing

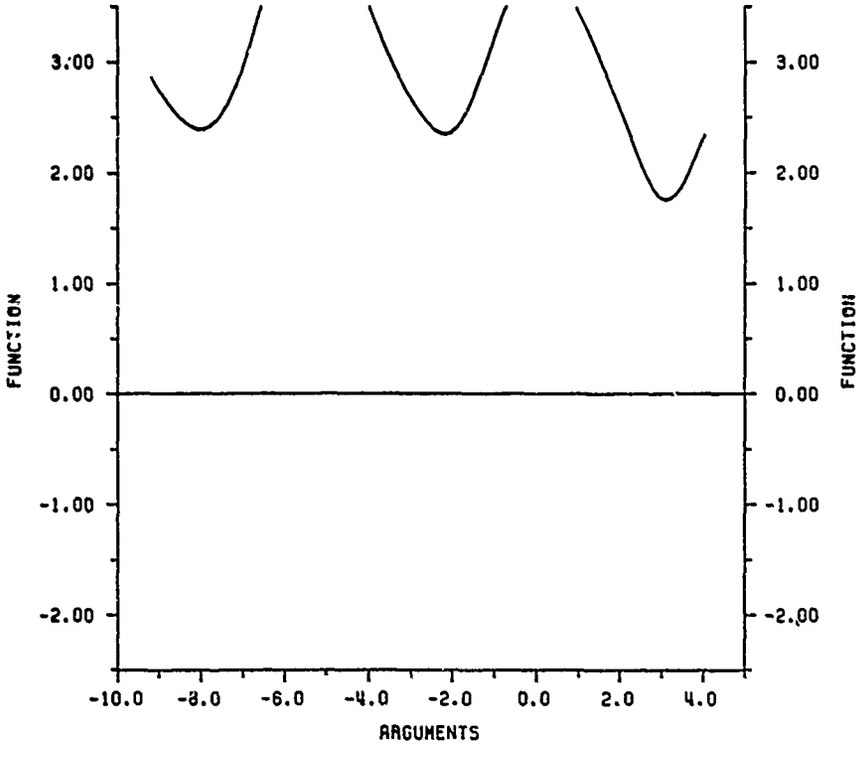


Figure 3. 1. 1c: Profile plot and clipping.

3.2 Multiple profiles

In the case that one or more profiles have to be compared with each other, it is advantageous to plot all of them in the same frame. In that case there is one natural restriction: the plotting window for all profiles have to be identical and to be known before the first profile is plotted. If no such plotting window has been defined, GSPP assumes the minima and maxima of the first $\{x_i\}$ and $\{y_i\}$ vectors to represent the window. However, in this case an unwanted clipping of further profiles (in the same frame) occurs whenever it exceeds the x- and y- minima/maxima of the first profile.

In order to simplify the identification of the different profiles, two additional features (relative to the single profile plot) have been built in: the profiles can be marked by centered symbols, each profile by a different one; the number of symbols relative to the number of profile data points can be selected by the user. Furthermore, there will be a legend plotted at the right end of the profile which consists of a list of symbol-input sequence identifications. The symbol plot along the profiles and the corresponding identification plot can be suppressed if so desired. Figures 3.2.1a, b show multiple profile plots without and with symbol suppressing.

The linewidth can be changed from profile to profile. Also the derivative can be changed from profile to profile.

All other optional procedures like axes plots, title and label plots, etc., are identical to the single profile case.

3.3 Implicitly defined profiles

A profile derived from a surface is herein called an implicitly defined profile. The surface is assumed to be a bicubic spline surface defined on a regular rectangular grid. If the data are not regularly distributed, the spline surface will be predicted first. Of course, this would not be necessary if only a single profile is calculated; usually

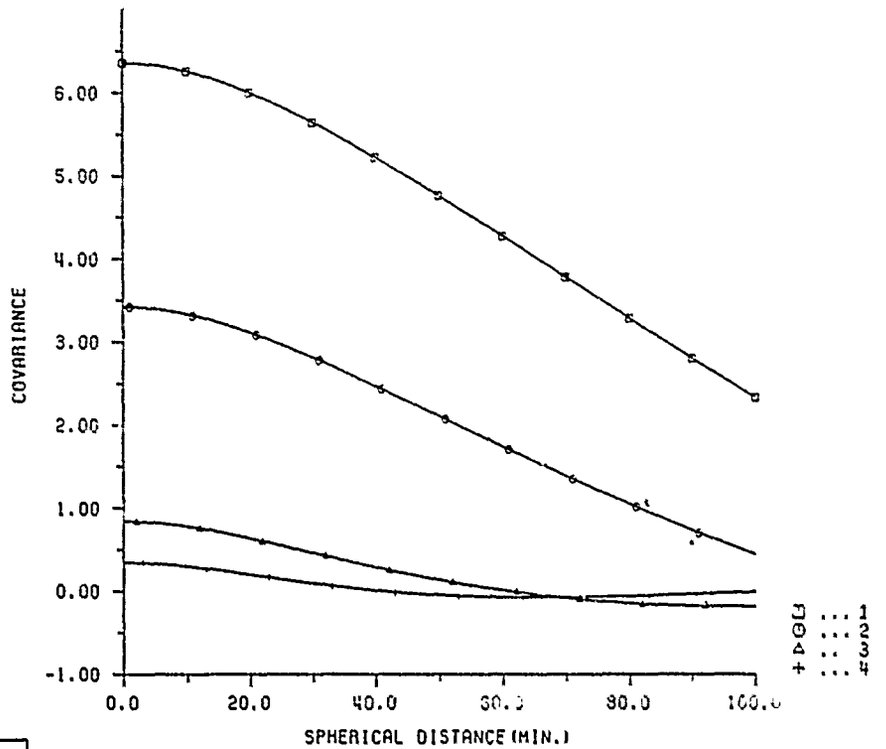
COVARIANCE FUNCTION FOR GEOIDAL HEIGHTS, MODEL T2

A=607.0
S=0.998444

*1 ... N= 36 DEG.VAR. SUBTRACTED
*2 ... N= 50 DEG.VAR. SUBTRACTED
*3 ... N=100 DEG.VAR. SUBTRACTED
*4 ... N=150 DEG.VAR. SUBTRACTED

Figure 3.2.1a:

Multiple profile plot.



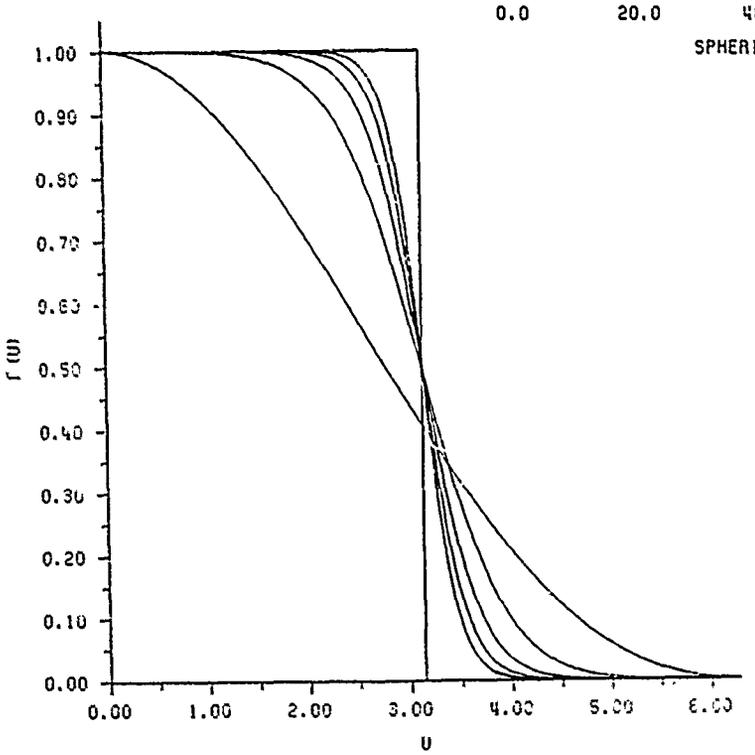


Figure 3.2.1b: Multiple profile plot; identification symbols and legend suppressed.

one needs a number of them and this is why we have decided to predict the whole surface. The prediction processes are identical with the ones described in Section 2.2, the spline representation is described in Section 2.4.2. The other essential information is the start- and end- point coordinates of the profile in consideration.

The profile needs not necessarily be a surface profile -- it can be a profile of any derivative of the surface

$$\frac{\partial^{\alpha} f(x,y)}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \quad , \qquad \alpha \leq 4 \; ; \quad \alpha_1 \, , \; \alpha_2 \leq 2 \, , \; \alpha = \alpha_1 + \alpha_2 \tag{3.3-1}$$

or also a profile of

$$\frac{\partial^{\alpha} f(x,y)}{\partial s^{\alpha}} \quad , \qquad \alpha \leq 2 \tag{3.3-2}$$

with ds the line element of the straight line connecting start- and endpoint of the profile. All derivatives (3.3-1) are discussed in Section 2.6.3; the derivatives (3.3-2) are simply the projection of the surface gradient onto the unit vector e with direction P_1-P_2 (P_1... start point, P_2... endpoint), $e^T = (\cos A, \; \sin A)$

$$\frac{\partial f}{\partial s} = \nabla f \cdot e \; = f_x \cos A + f_y \sin A, \tag{3.3-3}$$

and for the second order derivative we obtain

$$\frac{\partial^2 f}{\partial s^2} = e^T F e \tag{3.3-4}$$

with the second order tensor

$$F_{ij} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} \quad ;$$

then (3.3-4) has the form

$$\frac{\partial^2 f}{\partial s^2} = f_{xx}\cos^2 A + 2f_{xy}\cos A \sin A + f_{yy}\sin^2 A \ . \qquad (3.3\text{-}4)$$

Since the bicubic spline is twice continuously differentiable with respect to x and y, the derivative $\partial f/\partial s$ is continuous and has even a continuous derivative; $\partial^2 f/\partial s^2$ is only continuous.

The actual calculation of the profile is done pointwise along the line P_1-P_2 with a sampling rate of 2 points/mm; the profile is then a linear connection of all profile points.

3.3.1 Implicit profile procedures. There are a couple of features which are only connected with the implicit profile plots. First of all, the request for a smooth profile point interpolation will be ignored because the function itself is already smooth and the profile point sampling rate is sufficiently high; therefore, a smoothing would show up no difference to the linear point connection.

Since GSPP is capable of calculating and plotting 100 different profiles by just a single call, it is absolutely necessary to identify the different profiles. This is done automatically by a start- and endpoint message which appears at the bottom of the profile (see Figs. 3.3, 1 a, b). This message will always appear and will be centered unless its length exceeds the profile length even with minimal symbol height -- in this case it will be plotted left justified.

If the coordinates of the start- and/or endpoint are such that the profile happens to be outside the actual surface domain by 10% (this is the rectangular region on which the bicubic spline surface is defined), the plot will not be executed for this particular profile and an informative message will be edited on the line printer (or any other selected output unit).

Another difference for the explicit profile plots is the scaling of the horizontal axis. There are essentially two kinds of scaling offered by GSPP: a) the scaling is done according to the actual

distance P_1-P_2 with a tick mark interval which can be either chosen by the user or can be determined automatically;

b) a value 1.0 is assigned to the total length P_1-P_2 and the scaling is done accordingly.

A last difference is the grid plot: recall that a rectangular plotting window can be selected in the contour plot (see Section 2.8.4). When such a window has been defined and the profile runs within the window, crosses will be plotted at the intersections of grid lines (horizontal and vertical lines at tick mark interval). At the point where the profile crosses a window, a vertical bar will be plotted with the same linewidth as the profile itself; outside the window but inside the surface domain tiny crosses will be plotted instead of normal size crosses in order to indicate that this part of the profile runs within an area for which no contour plot has been performed. No profile and no crosses will be plotted if the profile leaves the surface domain up to the above mentioned 10% limit. Crosses will only be plotted below the profile. The following Figures 3.31(a, b, c) illustrate different kinds of profiles.

4. THREE-DIMENSIONAL SURFACE REPRESENTATIONS

A 3-D representation of a surface is, in this context, to be understood as a projection of a two-dimensional surface which is embedded in a three-dimensional Euclidean space, onto a plane. The plane can be arbitrarily oriented in space. The surface can be given either explicitly by function values at the grid points of a regular rectangular grid or by irregularly distributed data in which case a prediction algorithm (Section 2.2) takes care of the prediction of function values at all the grid points. Again, the surface is considered to be a bicubic spline surface defined by the function values at the grid points (see Section 2.6.2). The bicubic spline surface is then approximated by small bilinear elements; the function values at the

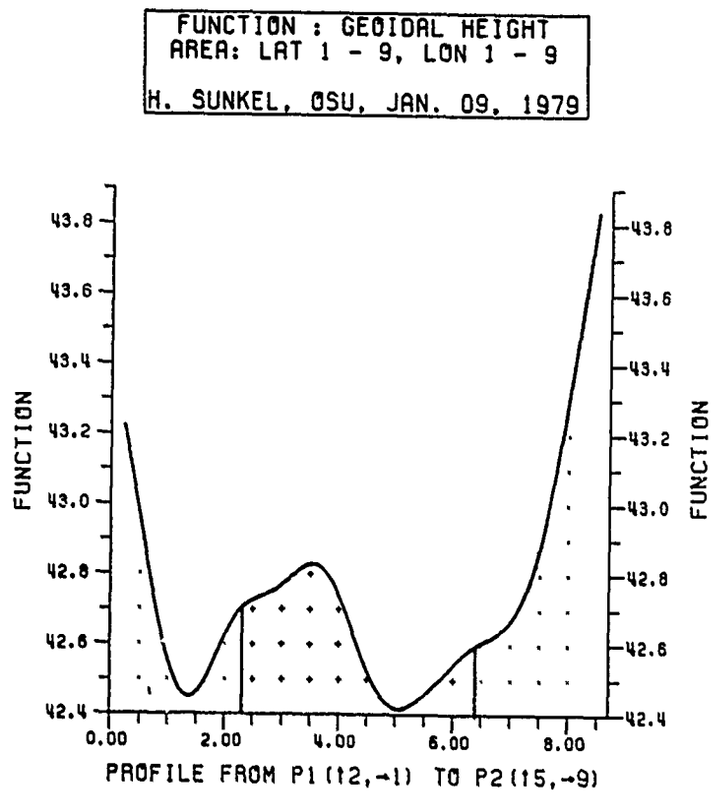


Figure 3.3.1a: Profile of a surface from P1 to P2 .

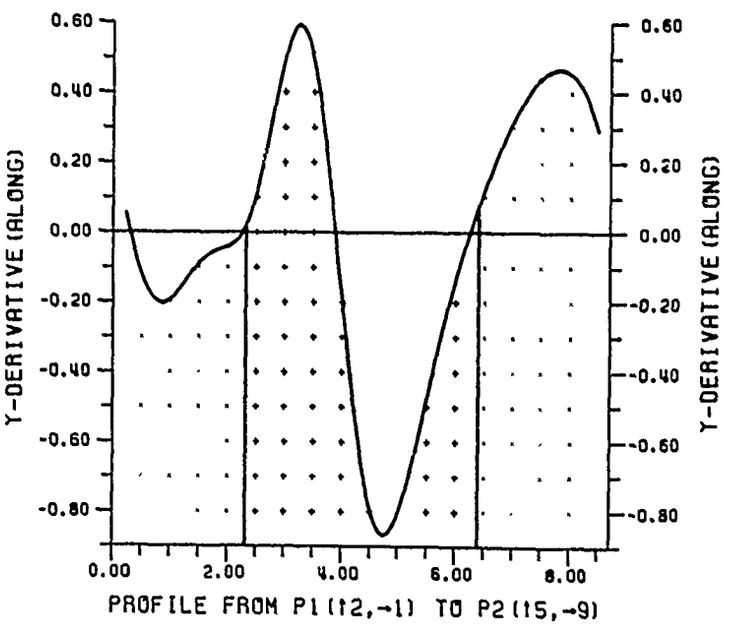


Figure 3.3.1b: Profile of the surface's D_y-derivative, from P1 to P2 .

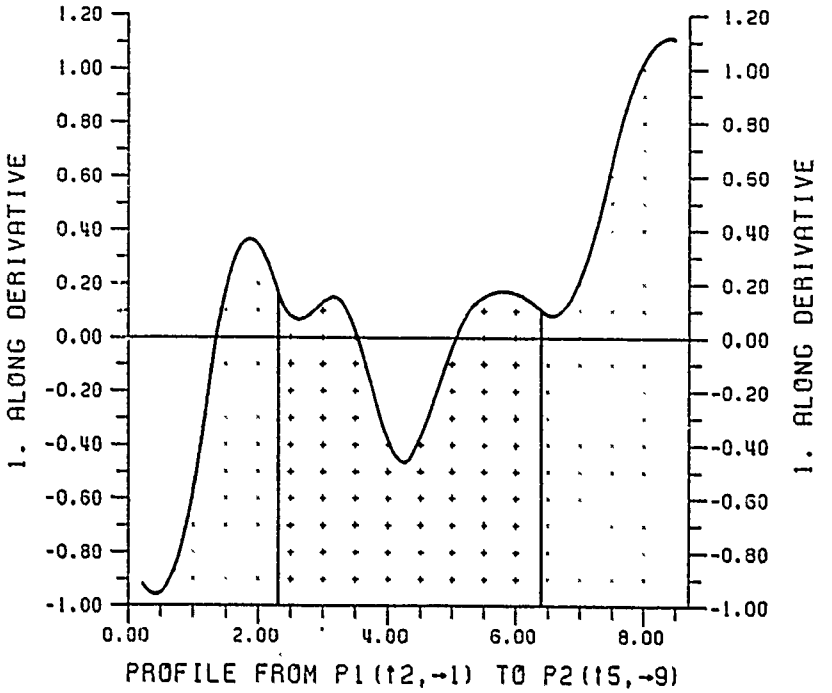


Figure 3.3.1c: Profile of the surface derivative in profile direction.

subgrid points (the grid on which the bilinear elements are defined) are interpolated spline-surface values. Up to this step the 3-D procedure does not differ from the contouring described in Chapter 2.

In the contouring procedure, these bilinear elements have to be intersected with a number of horizontal planes; in 3-D representations the bilinear elements are projected onto a plane with arbitrary orientation in 3-D space.

4.1 The projection

In principle, one could think of any kind of projection of the surface onto the plane. GSPP assumes that the projection is an axonometric projection, where the center of projection is located at infinity. Consequently, two parallels will remain parallel after the projection. (The projection equations used in GSPP could easily

be changed; therefore, it is, in principle, possible to obtain two perspective projections from two different centers of projection plotted in two complementary colors [red and green], which, viewed with anaglyph glasses, give a three-dimensional impression of the surface.)

The axonometric projection used in GSPP is defined in the following way: Assume the surface $z = z(x, y)$ to be defined on a rectangular domain D, with the lower left point of the rectangle coinciding with the origin (x=0, y=0), and the sides of the rectangle parallel to the coordinate lines. Let a plane pass through this origin. The orientation of the plane is defined by two angles, the longitude λ and the co-latitude θ in the following way: a coordinate system $(\bar{x}, \bar{y}, \bar{z})$ is associated with the plane with $\bar{x} = x$, $\bar{y} = y$, $\bar{z} = z$ if $\lambda = 0$ and $\theta = 0$. Let now the surface be fixed and let the plane rotate around the z-axis by the angle λ in the positive direction (looked upon from the origin, this is a clockwise rotation; looked upon from a point above the surface, this is a counter-clockwise rotation). The rotated coordinate system will be called (x', y', z')-system. Any point P with coordinates (x, y, z) will have coordinates

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

with

$$R_3 = \begin{bmatrix} \cos\lambda & \sin\lambda & 0 \\ -\sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The second rotation will be performed around the x'-axis by an angle θ in the positive direction (as defined above). The so obtained coordinate system is called $(\bar{x}, \bar{y}, \bar{z})$- system. In order to be clear, after the transformation, the two planes z=0 and $\bar{z} = 0$ span an angle θ with each

other. Any point P with coordinates (x, y, z) will, therefore, have coordinates

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} = R_1 R_3 \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

with

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} ,$$

and written explicitly ,

$$\begin{aligned} \bar{x} &= x\cos\lambda + y\sin\lambda \\ \bar{y} &= -x\cos\theta\sin\lambda + y\cos\theta\cos\lambda + z\sin\theta \\ \bar{z} &= x\sin\theta\sin\lambda - y\sin\theta\cos\lambda + z\cos\theta . \end{aligned} \qquad (4.1\text{-}1)$$

The next step is the projection of the point P, from the center of projection on the $\bar{z}$-axis at infinity, onto the $(\bar{x}, \bar{y})$-plane, which gives the Cartesian coordinates of the image point P* of P,

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = P_\perp \begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} \qquad (4.1\text{-}2)$$

with

$$P_\perp = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} .$$

Combining (4.1-1) and (4.1-2), replacing the co-latitude θ by the latitude $\varphi = 90 - \theta$, allowing a coordinate shift in the (x^*, y^*) - system and scale factors c_1 for (x, y) and c_2 for z we obtain the final projection equations

$$x^* = x_0 + (x \cos\lambda + y \sin\lambda)\, c_1$$

$$y^* = y_0^* + (-x \sin\varphi \sin\lambda + y \sin\varphi \cos\lambda)\, c_1 + z \cos\varphi\, c_2 \,. \qquad (4.1\text{-}3)$$

Another interpretation of (4.1-3), apart from the shift (x_0^*, y_0^*) and the scale factors, is the following: The surface domain is a rectangle in the equatorial plane with the x-axis through the origin and with orientation angles $\varphi = 0$ and $\lambda = 90^\circ$; the y-axis passes also through the origin with orientation angles $\varphi = 0$ and $\lambda = 180^\circ$; then the surface is, according to equation (4.1-3), looked upon from a point at infinity with coordinates φ and λ on the unit sphere.

4.2 Scale and shift

In a 3-D plot the user usually faces the difficulty of reducing the coordinates x, y and the corresponding function values z such that the figure "looks nice". Moreover, there is the problem of getting the plot on a particular place of the plotting sheet. This would make a number of decisions and calculations necessary if the data are already regularly distributed on the rectangular grid and if they are known to the user. He would first have to find the minimum and maximum of the function values, then make the necessary projections in order to find out the plotted size of the surface and so forth. If the data are irregularly distributed, probably heterogeneous and noisy, such a decision usually becomes a pure guess. GSPP is smart enough to do this job if the user wants it to be done, moreover, he can still make all or part of his decisions -- GSPP will accept them if they are consistent and reasonble, and will reject them and replace them by reasonable ones if they were unreasonable.

Let us briefly describe how GSPP finds scales and shift parameters: The scale factor c_1 is determined such that a 2-D plot would have an across the plot sheet extension of 10 cm. Since an axonometric projection reduces lengths (or keeps it constant), a square of 10 x 10 cm

never exceeds an along or across maximum of 20 cm. The next step consists in the calculation of the minima and maxima of the surface function values. These extreme values are projected using a simplified form of (4.1-3) and the scale factor c_2 for the function values is determined such that the projected difference of these extrema is 7.5 cm. If the values (0, 0) are assigned to the direction of view (λ, φ), GSPP will interpret them as "no input" values and will assign default values $\lambda = -60^{\circ}$, $\varphi = 30^{\circ}$ which usually gives a nice view of the surface. In order to find the actual along and across (the plot sheet) extensions, it is necessary to find the projected coordinates of the four corner points of the rectangle with minimal and maximal function values assigned to them. This gives obviously only upper bounds of the 3-D plot size which are then used to find the appropriate shift parameters (x_0^*, y_0^*). (The determination of the exact extension across the plot sheet would make the projection of all surface points necessary. This is a rather time-consuming task and should, therefore, be avoided.)

4.3 The 3-D plot

As soon as the bicubic spline surface is available, a piecewise bilinear approximation will be calculated by a simple interpolation procedure. The size of the bilinear elements can either be chosen by the user or, otherwise, will be selected by GSPP. The interpolation is done first row wise, then columnwise: the first along profile, a continuous and piecewise linear function is interpolated from the surface, is projected using (4.1-3) and plotted; then the next parallel profile is interpolated and projected. This profile and further profiles have to pass a hidden line algorithm which determines the actual visibility of the profile; those profiles or portions of profiles which are hidden by previous profiles are masked and will not be plotted. The hidden line algorithm used in GSPP is a modified version of that one described in

(Watkins, 1973). In this way all horizontal and vertical profiles of the surface are treated. After completion a frame is plotted which also passes the hidden line algorithm. The frame plot can be suppressed.

4.4 Additional and optional procedures

In any case , information about the viewing direction will be plotted in the lower left corner of the plot. This information gives the used values of the longitude and latitude of the direction of view in degrees and minutes.

A rectangle will be drawn around the entire plot; it can be suppressed.

At the top a title consisting of no more than 10 title lines can be plotted in the same way as for the contour and profile plots (with options: left justified, centered, right justified, or as on the input cards). The following Figures 4.4.1(a, b) show a contour plot and a corresponding 3-dimensional view.

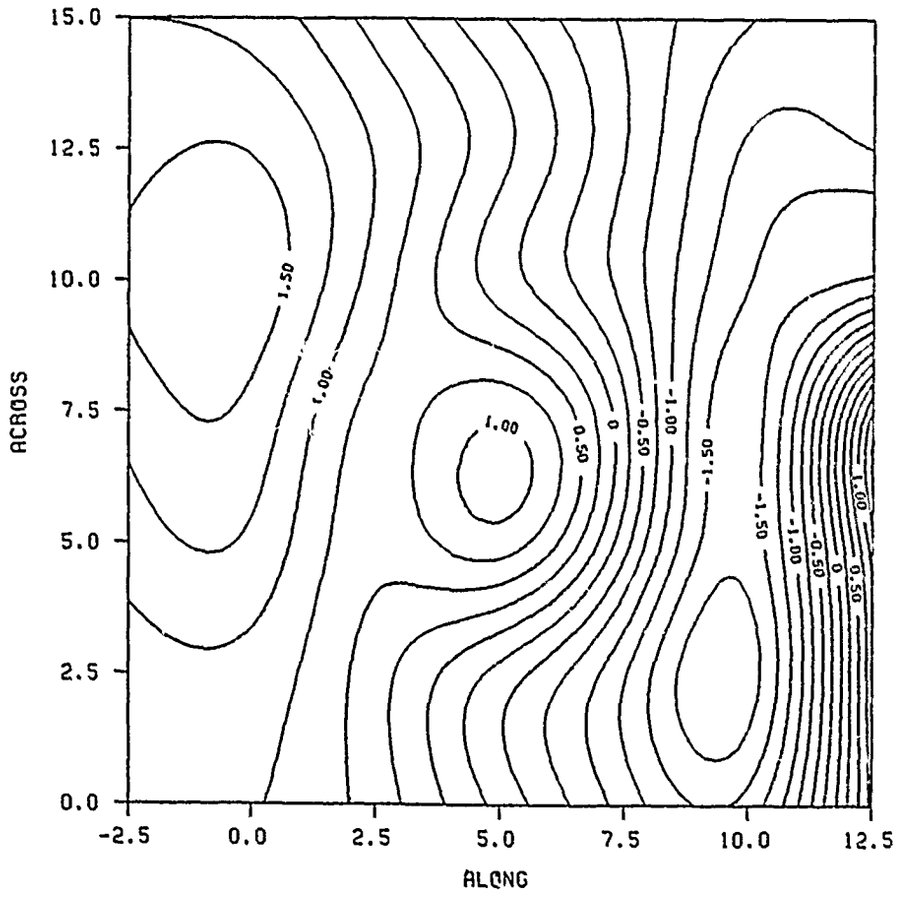


Figure 4.4.1a:

Contour plot

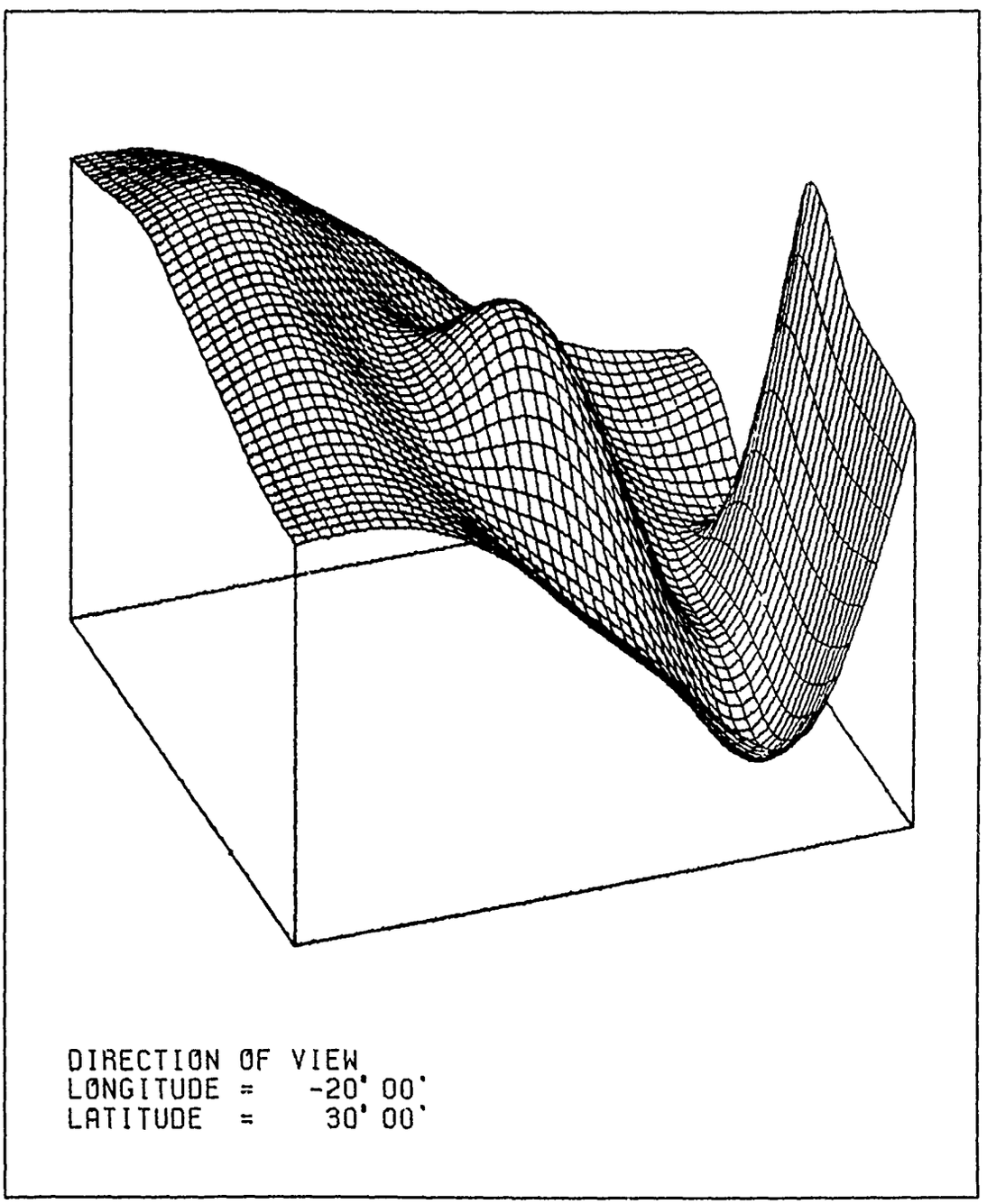


Figure 4.4.1b: 3-D plot corresponding to Fig. 4.4.1a

PART B

ISOLATED PROBLEMS

It could have been anticipated at the beginning of this report (and actually shown in PART A) that contouring, profile finding and drawing, and the generation of a 3-D plot consists of many procedures. The user of GSPP need not be confronted with the solution of these problems; however, many of these procedures (or subroutines) can be used in an isolated form and might be quite useful as such or as parts of other algorithms. Therefore, this section of the report discusses all the procedures and algorithms which are independent but integrated parts of GSPP.

1. Organization of data

As part of the contouring algorithm, the data sorting and retrieving procedure was briefly described in Section 2.1 of part A. By data organization we understand an algorithm which generates pointer vectors based on the two-dimensional distribution of the data. These pointer vectors should make a very fast retrieving of data within a specified array possible. The generation of the pointer vectors should take as little time as possible since a huge number of data will be organized. The following is a description of the method used by GSPP. The organization is performed by the subroutine OAF.

First, the working (or organization) domain, which is assumed to be a rectangle, has to be defined by its lower and upper x- and y-coordinates (the sides of the rectangle are assumed to be parallel to the coordinate lines). This rectangle is divided into M*N subrectangles of equal area (M in x-direction, N in y-direction). The program finds for each coordinate pair (x, y) the corresponding element (m, n). After a couple of operations (mainly integer additions and subtractions) four pointer or counter vectors are generated, IC1(.), ..., IC4(.):

The vector IC1(.) has a length equal to the number of data and contains, after completion, the element index i corresponding to each data. The element index i results from the two subrectangle

indices (m, n), associated with a data point (x, y), and is calculated according to

$$i = (m-1)N + n,$$

which means i increases rowwise if x is oriented northwards, i increases columnwise if x is oriented eastwards. Therefore, i = IC1(j) is the index of the subrectangle in which the data point (x_j, y_j) is located. If the data happens to be outside the working rectangle, an index M*N+1 will be assigned.

The vector IC3(.) is a counter vector of length M*N+1; IC3(i) equals the number of data in the subrectangle with index i. IC3(M*N+1) is equal to the number of data outside the rectangular working domain.

The vector IC4(.) is an auxiliary vector of length M*N+1 ; its elements are partial sums of the elements of IC3(.): IC4(1) = 1, IC4(k) = IC4(k-1) + IC3(k-1), k=2, ..., M*N+1.

Finally, the most important vector is IC2(.); it is organized in such a way that the first data in the subrectangle with index i has the original index IC2(IC4(i)); its length is equal to the number of data.

The data retrieving process runs then as follows. Assume one wants to know all data which are located within the element i: there are altogether IC3(i) data in this element; the index of the first data is IC2(IC4(i)), the index of the second data is IC2(IC4(i)+1) , and so forth; the last data has the index IC2(IC4(i) + IC3(i)-1). If there are no data within the element i, then IC3(i) = 0 and the index counting would be one step backward. Therefore, whenever there are no data within a particular element, the index retrieval described above does not apply.

The following example may illustrate the foregoing (Table 1.1 and Fig. 1.1). This kind of data organization is extremely fast because there are only very simple and very few operations involved; the data itself are not shifted -- they remain on their original storage locations.

```
C     A DATA ORGANIZATION EXAMPLE
C     SUBROUTINES USED : OAF
C     OUTPUT UNIT : 6
      COMMON /XDAT/X(30) /YDAT/Y(30) /USORPR/XMI,XMA,YMI,YMA,DUMMY(2),M,
     *       N /SORT1/IC1(30) /SORT2/IC2(30) /SORT3/IC3(13)
     *       /SORT4/IC4(13)
C     THE VECTORS IC1(.), ..., IC4(.)  NEED NOT HAVE THEIR PROPER DIM.
C     AS LONG AS THE NUMBER OF DATA IS LESS THAN OR EQUAL TO 1000 AND AS
C     LONG AS THE NUMBER OF SORT ELEMENTS IS LESS THAN OR EQUAL TO 1000
C
C     THE COORDINATES OF THE DATA POINTS HAVE TO BE STORED ON X(.) AND
C     Y(.). HERE WE GENERATE THESE DATA :
C     THE NUMBER OF DATA IS NDAT
      NDAT=30
      DO 1 I=1,NDAT
      X(I)=SIN(I*2.)+0.1
1     Y(I)=COS(I*3.)-0.1
C     DEFINE THE WORKING RECTANGLE
      XMI=-1.
      XMA=1.
      YMI=-1.
      YMA=1.
C     DEFINE THE NUMBER OF SUBRECTANGLES (SORT ELEMENTS) FOR THE DATA
C     ORGANIZATION
      M=3
      N=4
C     THE NUMBER OF ELEMENTS IS 12
C     CALL THE DATA ORGANIZATION SUBROUTINE OAF (ONE ARGUMENT = NUMBER
C     OF DATA)
      CALL OAF(NDAT)
C     PRINT THE RESULTS (INDEX, DATA, VECTORS IC1(.), ..., IC4(.) )
      WRITE(6,6000)
      MN1=M*N+1
      DO 2 I=1,NDAT
      IF(I.LE.MN1) WRITE(6,6001) I,X(I),Y(I),IC1(I),IC2(I),IC3(I),IC4(I)
      IF(I.GT.MN1) WRITE(6,6002) I,X(I),Y(I),IC1(I),IC2(I)
2     CONTINUE
6000  FORMAT(1H0,4X,' I    X(I)     Y(I)        IC1(I)  IC2(I)  IC3(I)  IC
     14(I)',//)
6001  FORMAT(1H ,4X,I2,2(2X,F6.2),3X,4I8)
6002  FORMAT(1H ,4X,I2,2(2X,F6.2),3X,2I8)
      STOP
      END
```

Program for Table 1.1

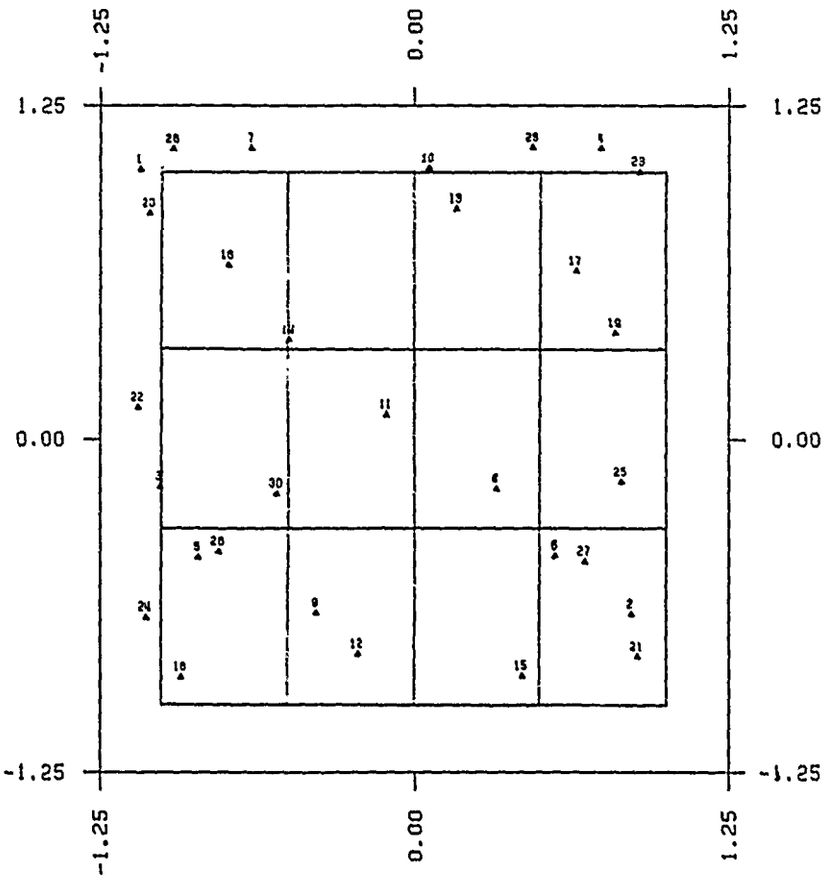


Figure 1.1:

Data distribution (position Δ and index).

I	X(I)	Y(I)	IC1(I)	IC2(I)	IC3(I)	IC4(I)
1	1.01	-1.09	13	5	3	1
2	-0.66	0.86	4	18	2	4
3	-0.18	-1.01	13	28	1	6
4	1.09	0.74	13	9	4	7
5	-0.44	-0.86	1	12	1	11
6	-0.44	0.56	4	15	1	12
7	1.09	-0.65	13	2	1	13
8	-0.19	0.32	7	6	1	14
9	-0.65	-0.39	2	21	1	15
10	1.01	0.05	13	27	1	16
11	0.09	-0.11	6	30	1	17
12	-0.81	-0.23	2	11	2	18
13	0.86	0.17	11	8	11	20
14	0.37	-0.50	10	25		
15	-0.89	0.43	3	16		
16	0.65	-0.74	9	14		
17	0.63	0.64	12	13		
18	-0.69	-0.93	1	17		
19	0.40	0.80	12	19		
20	0.85	-1.05	13	1		
21	-0.82	0.39	4	3		
22	0.12	-1.10	13	4		
23	1.00	0.89	13	7		
24	-0.67	-1.07	13	10		
25	-0.16	0.82	8	20		
26	1.09	-0.96	13	22		
27	-0.46	0.68	4	23		
28	-0.42	-0.78	1	24		
29	1.09	0.47	13	26		
30	-0.20	-0.55	5	29		

Table 1.1:

Data organization for data shown in Fig. 1.1.

2. Prediction

The basic principles of the prediction methods offered by GSPP have been shortly described in section 2.2 of PART A. In this chapter the practical aspects of least-squares prediction will be discussed.

The idea of least-squares collocation is to take into account all gravity field information, represented in terms of a heterogeneous data set, for the prediction of other gravity field quantities; in the ultimate case of collocation, model parameters can be incorporated into the solution.

This fine concept can hardly ever be fully realized in practice; reality demands sacrifices. The following facts make the unified solution a prohibitive task: covariance matrices are, in contrast to network normal equation matrices, full matrices. (In collocation one has to deal with the continuum "gravity field", in network problems with the discreteness of a continuum.) The size of the matrix depends, again in contrast to network problems, on the number of data. The more data we have, the better we have sampled the gravity field and -curiously enough- the more difficult it becomes to determine the gravity field: the instability of the covariance matrix increases with the data density since the equations become nearly linear dependent. Apart from the instability there is the problem to store the matrix - a very serious problem if more than a couple of thousand data are involved, not to speak about the actual inversion or calculation of the solution vector. Last, but by no means least, there remains the actual calculation of the signal (the gravity field quantity) together with its estimated error at a huge number of grid points -- we have to keep in mind what we actually want: the determination of a gravity field surface which is sufficiently well represented by an array of function values together with an appropriate interpolation function.

How can we overcome these problems, what are the consequences?

A widely applied and generally accepted practice is data selection. (If in trouble, select -- in analogy to Jeffreys recommendation "If in doubt, smooth".) It is an obvious and well understood fact that the data in the immediate neighborhood of the prediction point, in general, contributes the most and remote data very little to the prediction of the gravity field quantity. (This does not hold, e.g., for the prediction of geoidal heights from gravity anomalies.) Local samples are considered rather than regional or global ones. The number of data used for a prediction depends primarily on the problem (and sometimes on the personal taste). Rapp (1979) uses only a very few (around 5) altimeter data for the prediction (or rather interpolation) of an array of geoidal heights and in (Rapp, 1978) some 200 points for the recovery of mean gravity anomalies from altimeter data. Schwarz (1976) stresses the fact that neighborhood - data are the essential information and suggests a data selection. Lachapelle (1977) considers some one-to-two hundred gravity anomalies and deflections of the vertical for the combined solution collocation and integral formulas.

The consequences are as follows: local collocation solutions prevent an estimation of regional and global parameters like datum shifts for obvious reasons. Collocation in the local mode can only provide a less than optimal gravity field solution since an optimal solution would require all data to be taken into consideration. This is the price we pay for a gain in matrix stability, limitation of storage requirements and for keeping the computation time at an acceptable level.

In gravity field surface prediction, the factor time plays, apart from the others discussed above, a particularly important role. This is why only local solutions can be envisioned under the present circumstances.

The surface prediction algorithum of GSPP is designed for the local mode only. It considers up to 100 data in the neighborhood of the prediction point. (This number has been kept low because of the

storage limitations of the used computer; an increase to 200 or even 300 for a larger computer system requires only a few changes in the program.) In order to represent the gravity field surface sufficiently well, a high density of prediction points (grid points) has to be chosen. If not more than 100 data are found in the prediction region, the algorithm uses all the data, calculates the inverse of the corresponding covariance matrix and predicts the signals together with its rms-errors at all grid points. The situation changes if more than one hundred data are used. The algorithm switches over to a mode which can best be described as moving inverse covariance prediction. Its principle is as follows. Assume a fairly homogeneous data distribution and a grid as in Fig.B2.1. gravity field surface function values are to be predicted at all the grid points. Assume furthermore a circular region R_i centered at the grid point P_i and another circular region R_{i+1} centered at a neighbouring grid point P_{i+1} (see Fig. B2.1). All data within R_i are used for the prediction of S_i (the gravity field surface function value at the grid point P_i), all data within R_{i+1} are used for the prediction of S_{i+1}, have a common subset $S_{i,i+1}$ which is the intersection of S_i and S_{i+1},

$$S_{i,i+1} = S_i \cap S_{i+1} .$$

In Fig.B2.1 the subset $S_{1,2}$ consists of the data (black dots) within the crosshatched region.

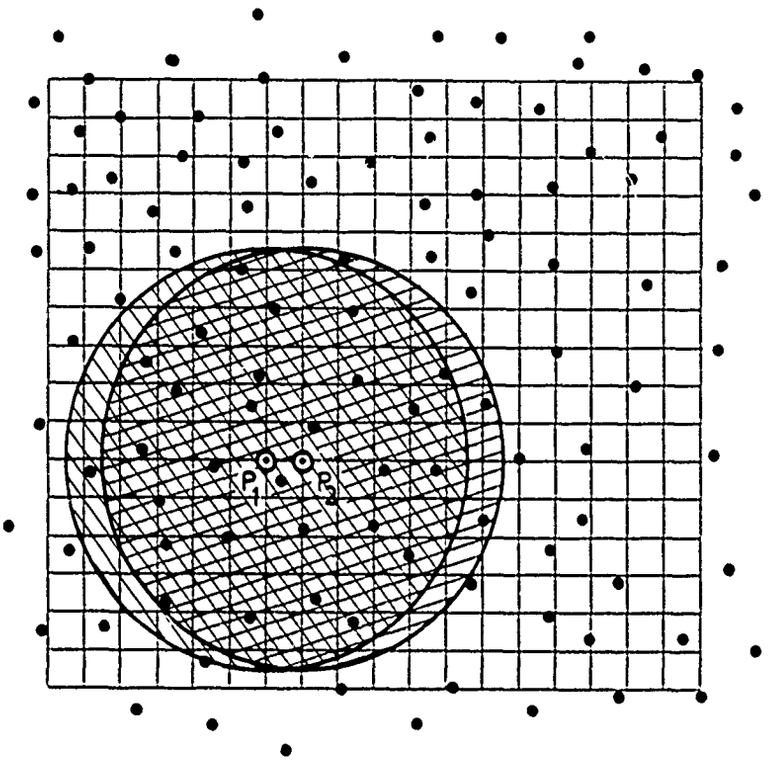


Figure B2.1: Data selection in moving inverse covariance prediction mode

Let N_i and N_{i+1} be the number of data belonging to the set S_i and S_{i+1}, respectively, and let $N_{i,i+1}$ be the number of data belonging to the common subset $S_{i,i+1}$. (In Figure B2.1, $N_1 = 28$, $N_2 = 30$, $N_{1,2} = 27$.) The corresponding covariance matrices are C_i and C_{i+1}, its common part $C_{i,i+1}$. If the grid is dense relative to the data distribution (a necessary requirement for surface prediction), then C_{i+1} will differ only slightly from C_i; with other words, the difference matrices C_i $C_{i,i+1}$ and $C_{i+1}-C_{i,i+1}$ will have only a few non-zero elements. (In Fig. B2.1 C_1 has 406, C_2 has 465, $C_{1,2}$ has 378;

therefore C_1-$C_{1,2}$ has only 28, C_2-$C_{1,2}$ only 87 in general non-zero elements. Here we have considered only the upper triangular part of the symmetric covariance matrix.) Consequentlv, it should be possible to find C_{i+1}^{-1} (the inverse covariance matrix corresponding to point P_{i+1}) in a fast and simple way, if C_i^{-1} is known. In fact, the problem reduces, in principle, to a two-fold application of matrix inversion by block-partitioning.

Let C_i be partitioned into 4 parts $C_{i,i+1}$; B_i, B_i^T, D_i and similarly its inverse C_i^{-1} into K_i, L_i, L_i^T, M_i,

$$C_i = \begin{pmatrix} C_{i,i+1} & B_i \\ B_i^T & D_i \end{pmatrix}, \qquad C_i^{-1} = \begin{pmatrix} K_i & L_i \\ L_i^T & M_i \end{pmatrix} \tag{2.1}$$

There it follows from simple matrix algebra (Faddejew - Faddejewa, 1970, p. 201 ff.), that

$$M_i = (D_i - B_i^T \; C_{i,i+1}^{-1} B_i)^{-1}, \tag{2.2a}$$

$$L_i = -C_{i,i+1}^{-1} B_i M_i, \tag{2.2b}$$

$$K_i = C_{i,i+1}^{-1} (I - B_i L_i^T) \tag{2.2c}$$

with I denoting the unit matrix. In the same way as above, the matrices C_{i+1} and C_{i+1}^{-1} can be partitioned.

$$C_{i+1} = \begin{pmatrix} C_{i,i+1} & B_{i+1} \\ B_{i+1}^T & D_{i+1} \end{pmatrix}, \qquad C_{i+1}^{-1} = \begin{pmatrix} K_{i+1} & L_{i+1} \\ L_{i+1}^T & M_{i+1} \end{pmatrix} \tag{2.3}$$

Note that $C_{i,i+1}$ is common to both, C_i and C_{i+1} and so is the inverse $C_{i,i+1}^{-1}$. If this common inverse is known, then C_{i+1}^{-1} can be found in the way described above,

$$M_{i+1} = (D_{i+1} - B_{i+1}^T C_{i,i+1}^{-1} B_{i+1})^{-1} \qquad (2.3a)$$

$$L_{i+1} = -C_{i,i+1}^{-1} B_{i+1} M_{i+1} , \qquad (2.3b)$$

$$K_{i+1} = C_{i,i+1}^{-1} (I - B_{i+1} L_{i+1}^T) . \qquad (2.3c)$$

Therefore, the problem of calculating C_{i+1}^{-1}, if C_i^{-1} is known, consists in finding $C_{i,i+1}^{-1}$ and in calculating M_{i+1}, L_{i+1}, K_{i+1} following the equations above.

Multiplying equation (2.2b) by $M_i^{-1} L_i^T$, we obtain

$$L_i M_i^{-1} L_i^T = -C_{i,i+1}^{-1} B_i L_i^T,$$

which gives, by inserting into (2.2c), the inverse of the covariance matrix corresponding to the common data set $S_{i,i+1}$, expressed by the known four submatrices of C_i^{-1},

$$C_{i,i+1}^{-1} = K_i - L_i M_i L_i^T . \qquad (2.4)$$

The important point is that M_i as well as M_{i+1} is usually very small compared with $C_{i,i+1}$; (for the data configuration of Figure 2.1, M_1 has dimension (1, 1), M_2 has dimension (3, 3) and $C_{1,2}$ has dimension (27, 27).) It is important to realize that, for the transition from C_i^{-1} to C_{i+1}^{-1}, only two inversions of generally very small matrices with dimensions of M_i and M_{i+1} are necessary. (The vector and matrix multiplication are relatively inexpensive.)

There needs still one problem to be solved which was tacitly passed by: the matrix C_i^{-1} has to be re-ordered according to a permintation vector whose elements point to the data within the prediction circle. This procedure consists mainly of logical operations and is very fast; it is accomplished by the subroutines BUBBLE (which makes a vector bubble sort,

generating the permintation vector), and MATUMD (which does the actual matrix reordering).

The principle is as follows: N_i data were found in the region S_i, a vector V_i is stored; its elements point to the data in S_i, the inverse covariance matrix C_i^{-1} is given; in the next step N_{i+1} data are found in the region S_{i+1}, a vector V_{i+1} is generated whose elements point to the data in S_{i+1}. $N_{i,i+1}$ elements are common to V_i and V_{i+1}, but the corresponding sequential locations within V_i and V_{i+1} will, in general differ. Therefore, V_{i+1} passes a sorting algorithm which "bubbles" all common $N_{i,i+1}$ data in V_{i+1} upward such that, after the bubble sort, the $N_{i,i+1}$ common elements occupy the first $N_{i,i+1}$ places in the vector V_{i+1}. A pointer vector W_i is generated whose elements point to the retained elements in V_i.

j	$V_i(j)$	$V_{i+1}(j)$ before sort	$V_{i+1}(j)$ after sort	$W_i(j)$
1	3	7	3	1
2	17	8	8	3
3	8	79	24	4
4	24	3	45	6
5	11	41	7	2
6	45	24	79	5
7		45	41	
8		13	13	

Table 2.1 Example of bubble sort used in the prediction part of GSPP

This vector W_i serves as a permutation vector for the re-ordering of the inverse covariance matrix C_i^{-1}.

At this point the reader might ask why we are calculating the inverse covariance matrix; we could probably apply a similar algorithm for the calculation of the solution vector of the linear system.

$$C\alpha = C_p \tag{2.5}$$

with C...covariance matrix,

C_p...cross-covariance vector (signal-data)

α...solution vector

Note that the right side of equation (2.5) is C_p and not the data vector l; this can be done since we estimate for each solution vector only one signal. The predicted signal is then given by

$$S_p = \alpha^T l \tag{2.6a}$$

and its estimated error variance by

$$m_p^2 = C_{pp} - \alpha^T C_p \tag{2.6b}$$

This method is reportedly two to three times faster (Lachapelle, 1977) essentially because it bypasses the matrix inversion. Here the inversion method has been chosen since it looked more transparent to the author; the solution vector method should be investigated.

As mentioned before, the use of the moving inverse prediction method is somewhat restricted; it is very advantageous for the solution of very large problems which reduce essentially to interpolations, differentiations and/or downward continuatious; examples are: determination of a digital geoid based on altimeter data, solution of the Bjerhammar problem, prediction of mean gravity anomalies from point gravity anomalies, gravity interpolation, interpolation of vertical deflections, determination of mean gravity anomalies from gravity and gradiometer data, etc. For the solution of problems which involve the whole data vector and allow, in addition, the estimation of parameters, an excellent operational system is available which is based on stepwise least-squares collocation (Tscherning, 1974).

3. Regression

The calculation of the parameters of a least-squares regression polynomial, in GSPP, is based on the following premises: the data are error-free and homogeneous evaluation functionals of a surface; data (and surface) are defined on the Euclidean plane. Then a least-squares regression polynomial solution, based on the data $\{x_i\}$, $\{y_i\}$, $\{f_i\}$, i= 1, ..., I, $f_i = f(x_i, y_i)$, is described in Section 2.3 of Part A; the parameters of the polynomial are given by equation (2.3-2). The design matrix $\Phi = \{\Phi_{ij}\} = \{L_i \varphi_j\}$, φ_j ... base-functions, in its explicite form , is given by

$$\Phi = \begin{bmatrix} 1, & x_1, & y_1, & x_1^2, & x_1y_1, & y_1^2, & \ldots, & y_1^n \\ 1, & x_2, & y_2, & x_2^2, & x_2y_2, & y_2^2, & \ldots, & y_2^n \\ \vdots & & & & & & & \vdots \\ 1, & x_I, & y_I, & x_I^2, & x_Iy_I, & y_I^2, & \ldots, & y_I^n \end{bmatrix} .$$

The corresponding normal equation matrix $\Phi^T\Phi$ (for equal weights) is unstable for large n. GSPP allows the degree n to vary between 0 and 5; the degree has to be such that the number of data I is bigger than the number of parameters J, with J = J(n) given by

$$J = \frac{(n+1)\,(n+2)}{2} \quad ;$$

there is no generalized inverse solution allowed in GSPP. If n has been defined such that $I \leq J$, the highest possible degree will be chosen. For reasons of stability, the coordinates $\{x_i, y_i\}$ are transformed linearly such that $\min_{\forall i}(x_i', y_i') = 0$ and $\max_{\forall i}(x_i' \text{ or } y_i') = 1$, depending on whether the range of x or y is bigger. Therefore, also the parameters $\{a_j\}$, j = 1, ..., J, refer to these transformed coordinates.

This has to be taken into account if operations of any kind are to be applied on the polynomial.

The actual calculation of the least-squares regression polynomial is performed in the subroutine REGPOL. If the domain of definition was not specified by the user, it will be defined in GSPP (if REGPOL is run separated from GSPP, these values have to be defined). The output is the vector of polynomial coefficients, normalized as described above, the root mean square and average absolute approximation error, the individual approximation errors (actual function values minus polynomial derived function values) if requested, and a matrix of polynomial derived function values at the grid points of a user-specified (or GSPP selected) regular rectangular grid. These grid point values, in turn, can be used by GSPP for profiling, contouring and a 3-D plot.

In the following we give some examples of how the regression part can be used.

3.1 Regression polynomial based on irregularly distributed data

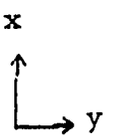
In this section a typical application of a least-squares regression is shown: there is given a set of error-free, homogeneous, irregularly distributed data defined on the two-dimensional Euclidean plane. A least-squares regression polynomial of a certain degree has to be calculated and interpolated at the grid points of a regular rectangular grid. In addition, the individual approximation errors (data reproduction errors) should be calculated. In the following program the polynomial degree has been chosen to be equal to 4 which is too high relative to the number of data which was selected to be equal to 14. Therefore, the program changes the degree to 3.

In the sequel the program plus input/output are listed.

```
C       AN EXAMPLE OF A LEAST-SQUARES REGRESSION POLYNOMIALBASED ON
C       IRREGULARLY DISTRIBUTED DATA
        DIMENSION A(8,8,1), ZZ(14)
        COMMON /XDAT/X(14) /YDAT/Y(14) /ZDAT/Z(14) /UCIP/IDUM1,MX,NY,
       *      IDUM2(6),IW1,IDUM3(17),N /UCRP/XLL,YLL,DX,DY /IOSPAR/IOUT
C       A STORES THE POLYNOMIAL INTERPOLATED FUNCTION VALUES AT THE GRID
C       POINTS
C       NDAT IS THE NUMBER OF DATA
C       IRR ... =0 OR .GE.0 ... DATA ARE IRREGULARLY DISTRIBUTED
C       N ... DEGREE OF POLYNOMIAL
C       X(.), Y(.) ... DATA LOCATION COORDINATES (X ACROSS(SOUTH - NORTH),
C       Y ALONG (WEST - EAST))
C       Z=Z(X,Y) ... FUNCTION VALUES
C       IW1 ... =1 OR 2 ... ALSO CALCULATION OF INDIVIDUAL APPROXIMATION
C       ERRORS; IN THIS CASE THE ORIGINAL FUNCTION VALUES WILL BE DES-
C       TROYED AND REPLACED BY THE INDIVIDUAL DATA REPRODUCTION ERRORS
C       MXMAX, NYMAX, MZMAX ARE THE DIMENSIONS OF A AS DEFINED IN THIS
C       PROGRAM
C       MX, NY ... NUMBER OF GRID POINTS IN X AND Y DIRECTION
C       MX.LE.MXMAX, NY.LE.NYMAX
C       XLL, YLL ... COORDINATES OF THE GRID'S LOWER LEFT CORNER (CORRES-
C       PONDS TO A(1,1,1))
C       DX, DY ... GRID DISTANCES IN X AND Y DIRECTION
C       IOUT ... OUTPUT UNIT FOR MESSAGES AND RESULTS
C       IDUM1, IDUM2(6) AND IDUM3(17) ARE DUMMY VECTORS
        NDAT=14
        IRR=0
        N=4
        IW1=1
        MXMAX=8
        NYMAX=8
        MZMAX=1
        MX=6
        NY=8
        XLL=90.
        YLL=-120.
        DX=5.
        DY=7.
        IOUT=6
C       READ THE DATA (HERE : GENERATE THE DATA)
        DO 1 I=1,NDAT
        X(I)=SIN(I*2.)*10+100
        Y(I)=COS(I*3.)*20-100
        Z(I)=COS(I*0.5)*SIN(I*0.9)
1       ZZ(I)=Z(I)
C       CALL THE SUBROUTINE REGPOL
        CALL REGPOL(A,MXMAX,NYMAX,MZMAX,NDAT,IRR)
C       PRINT THE DATA AND RESULTS
        WRITE(IOUT,6000)
        DO 2 I=1,NDAT
2       WRITE(IOUT,6001) I,X(I),Y(I),ZZ(I),Z(I)
        WRITE(IOUT,6002)
        DO 3 I=1,MX
        I1=MX-I+1
3       WRITE(IOUT,6003) (A(I1,J,1),J=1,NY)
6000    FORMAT('1',4X,' I          X(I)     Y(I)          Z(I)    DZ(I)',//)
6001    FORMAT(1H ,4X,I2,2(5X,2F8.2))
6002    FORMAT('1',1H)
6003    FORMAT('0',4X,8F8.2)
        STOP
        END
```

Program corresponding to Table 3.1.1.

I	X(I)	Y(I)	Z(I)	DZ(I)
1	109.09	-119.80	0.69	0.18
2	92.43	-80.89	0.53	-0.13
3	97.21	-118.22	0.06	-0.43
4	109.89	-83.12	0.13	-0.08
5	94.56	-115.19	0.73	0.40
6	94.63	-86.79	0.77	0.53
7	109.91	-110.95	-0.02	-0.33
8	97.12	-91.52	-0.52	-0.49
9	92.49	-105.84	-0.20	0.08
10	109.13	-96.91	0.12	0.43
11	99.91	-100.27	-0.32	-0.08
12	90.94	-102.56	-0.94	-0.20
13	107.63	-94.67	-0.74	-0.27
14	102.71	-108.00	0.06	0.30



0.73	0.57	0.34	0.10	-0.09	-0.19	-0.15	0.08
0.10	-0.04	-0.22	-0.40	-0.52	-0.54	-0.40	-0.06
0.15	0.01	-0.17	-0.33	-0.42	-0.39	-0.20	0.21
0.47	0.28	0.07	-0.11	-0.20	-0.17	0.04	0.48
0.63	0.36	0.08	-0.15	-0.30	-0.30	-0.10	0.33
0.22	-0.17	-0.56	-0.89	-1.12	-1.19	-1.05	-0.66

Table 3.1.1: Third degree least-squares regression polynomial based on irregularly distributed data x; y; z = z(x, y)

3.2 Regression polynomial based on regularly distributed data

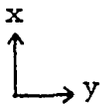
If data are distributed on a regular rectangular grid, and stored on an array corresponding to this grid, then a slightly modified version of the above listed program is necessary in order to obtain the regression field. The following example may serve as an illustration.

Both regressions described above can also be obtained by calling GSPP; the regression surface can be contoured, profiled or plotted as a 3-D view. These integrated applications will be described in PART C.

```
C     REGRESSION POLYNOMIAL BASED ON REGULARLY DISTRIBUTED DATA
      DIMENSION A(8,8,2)
      COMMON /UCIP/IDUM1,MX,NY,IDUM2(6),IW1,IDUM3(17),N /UCRP/XLL,YLL,DX
     *      ,DY /IOSPAR/IOUT
      NDAT=0
      IRR=-1
      N=5
      IW1=1
      MXMAX=8
      NYMAX=8
      MZMAX=2
      MX=6
      NY=8
      XLL=90.
      YLL=-120.
      DX=4.
      DY=5.
      IOUT=6
C     READ THE DATA; HERE: GENERATE THE DATA
      DO 1 I=1,MX
      DO 1 J=1,NY
1     A(I,J,1)=(SIN(I*2.)*COS(J*3.))*10
C     PRINT THE DATA
      DO 4 I=1,MX
      I1=MX-I+1
4     WRITE(IOUT,6003) (A(I1,J,1),J=1,NY)
C     CALL THE SUBROUTINE REGPOL
      CALL REGPOL(A,MXMAX,NYMAX,MZMAX,NDAT,IRR)
C     PRINT THE GRID VALUES OF THE REGRESSION POLYNOMIALS AND THE INDI-
C     VIDUAL APPROXIMATION ERRORS
      DO 2 I=1,MX
      I1=MX-I+1
      WRITE(IOUT,6003) (A(I1,J,1),J=1,NY)
2     CONTINUE
      DO 3 I=1,MX
      I1=MX-I+1
      WRITE(IOUT,6003) (A(I1,J,2),J=1,NY)
3     CONTINUE
6003  FORMAT(1H0,4X,8F8.2)
      STOP
      END
```

Program corresponding to Table 3.2.1

5.31	-5.15	4.89	-4.53	4.08	-3.54	2.94	-2.28
5.39	-5.22	4.96	-4.59	4.13	-3.59	2.98	-2.31
-9.79	9.50	-9.01	8.35	-7.52	6.53	-5.42	4.20
2.77	-2.68	2.55	-2.36	2.12	-1.85	1.53	-1.19
7.49	-7.27	6.90	-6.39	5.75	-5.00	4.15	-3.21
-9.00	8.73	-8.28	7.67	-6.91	6.00	-4.98	3.86



Regularly distributed data

AVERAGE ABSOLUTE POLYNOMIAL APPROXIMATION ERROR ... 4.804
RMS POLYNOMIAL APPROXIMATION ERROR ... 5.433

4.06	-0.55	-1.02	-0.40	0.00	-0.05	-0.39	-1.11
1.71	-0.48	-0.08	0.64	0.86	0.38	-0.09	-1.54
-1.53	-2.12	-1.25	-0.44	0.02	0.36	0.68	0.26
0.25	0.01	0.16	0.01	-0.21	-0.03	0.66	0.86
2.19	2.39	1.75	0.37	-0.97	-1.31	-1.14	-1.12
-4.22	-0.83	0.49	0.24	-0.30	-0.07	1.04	1.35

4th degree regression polynomial grid point values

1.25	-4.60	5.91	-4.13	4.07	-3.49	3.33	-1.17
3.67	-4.74	5.04	-5.23	3.27	-4.17	3.07	-0.76
-8.26	11.62	-7.76	8.79	-7.54	6.17	-6.10	3.94
2.52	-2.69	2.38	-2.37	2.33	-1.82	0.87	-2.64
5.30	-9.66	5.15	-6.75	6.72	-3.49	3.28	-2.09
-4.18	9.61	-8.78	7.44	-6.60	6.08	-6.02	2.31

Residuals

Table 3.2.1: Least-squares regression polynomial based on regularly distributed data.

4. Smooth surface representations

One essential part of GSPP is a set of subroutines which is needed for a smooth representation of a surface by a bicubic spline function together with a surface interpolation/differentiation algorithm.

4.1 Calculation of spline defining values

This section gives a practical example of how to obtain the spline defining values by the subroutine BISP, isolated from GSPP. The necessary background and formulas are contained in Section 2.6.2. of PART A.

Let us assume that function values are given at the grid points of a regular rectangular grid (m=1, ..., M; n=1, ..., N). Then the bicubic spline defining values is the set $\{f_{mn}, p_{mn}, q_{mn}, r_{mn}\}$, m=1, ..., M; n=1, ..., N (see Section 2.6.2) of PART A. The function values at the grid points $\{f_{mn}\}$ are assumed to be known, the derivatives $\{p_{mn}, q_{mn}, r_{mn}\}$ have to be determined. Under side conditions as explained in Section 2.6.2, of PART A this set of values defines the bicubic spline surface uniquely.

The following program can be used to calculate these values by using BISP.

```
C     AN EXAMPLE OF A BICUBIC SPLINE DEFINING VALUE CALCULATION
      DIMENSION A(10,10,4)
C     A(.,.,1) ... STORES THE REGULARLY DISTRIBUTED DATA
C     A(.,.,2) ... 1. DERIVATIVES IN X-DIRECTION (ACROSS, SOUTH - NORTH)
C     A(.,.,3) ... 1. DERIVATIVES IN Y-DIRECTION (ALONG, WEST - EAST)
C     A(.,.,4) ... 2. MIXED XY-DERIVATIVES
C     MX, MY ... ACTUAL DIMENSION OF THE GRID
C     OUTPUT UNIT = 6
      MX=6
      MY=5
      IOUT=6
C     READ THE DATA; HERE: GENERATE THE DATA
      DO 1 I=1,MX
      DO 1 J=1,MY
1     A(I,J,1)=(SIN(0.5*I)+COS(0.6*J))*10
C     CALCULATE THE DEFINING VALUES
      CALL BISP(A,10,10,MX,MY)
C     PRINT DATA AND RESULTS
      DO 3 K=1,4
      WRITE(IOUT,6000)
      DO 2 I=1,MX
      I1=MX-I+1
2     WRITE(IOUT,6001) (A(I1,J,K),J=1,MY)
3     CONTINUE
6000  FORMAT(1H0,1X,//)
6001  FORMAT(1H0,4X,5F10.4)
      STOP
      END
```

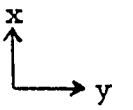
Program to Table 4.1.1.

Table 4.1.1

Bicubic spline defining values

9.6646	5.0348	-0.2608	-5.9627	-8.4887
14.2381	9.6083	3.7127	-1.3892	-3.9152
17.3463	12.7166	6.8210	1.7190	-0.8070
18.2280	13.5985	7.7029	2.6010	0.0750
16.6681	12.0386	6.1427	1.0408	-1.4852
13.0476	8.4178	2.5222	-2.5797	-5.1057

Function values at the grid points (data)



-4.8432	-4.8432	-4.8432	-4.8432	-4.8432
-4.0841	-4.0841	-4.0841	-4.0841	-4.0841
-2.0657	-2.0657	-2.0657	-2.0657	-2.0657
0.3262	0.3262	0.3262	0.3262	0.3262
2.7958	2.7958	2.7958	2.7958	2.7958
4.0328	4.0328	4.0328	4.0328	4.0328

D_x-derivatives

-4.2800	-5.3293	-5.9790	-3.7474	-1.9153
-4.2800	-5.3293	-5.9790	-3.7474	-1.9153
-4.2800	-5.3293	-5.9790	-3.7474	-1.9153
-4.2800	-5.3293	-5.9790	-3.7474	-1.9153
-4.2800	-5.3293	-5.9790	-3.7474	-1.9153
-4.2800	-5.3293	-5.9790	-3.7474	-1.9153

D_y-derivatives

-0.0000	0.0000	0.0000	-0.0000	-0.0000
-0.0000	-0.0000	0.0000	-0.0000	0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000
0.0000	0.0000	0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000	0.0000
0.0000	0.0000	-0.0000	0.0000	0.0000

D_{xy}-derivatives

The reader will probably realize that no grid distances have been defined. This is not necessary because the defining values returned from BISP are normalized ones: they refer to a square grid of grid distance equal to 1. The reasons for the normalized calculation are simplicity and stability; there is no restriction of generality involved: it can easily be shown that the defining values referring to a non-normalized grid can simply be obtained by dividing the x-derivatives $\{p_{mn}\}$ by the grid distance in x-direction, the y-derivatives $\{q_{mn}\}$ by the grid distance in y-direction and the second mixed derivatives $\{r_{mn}\}$ by the product of x- and y- grid distances.

BISP is designed for a maximum grid size of 300 x 300. The CPU-time needed increases only linearly with the number of grid points involved, a good rule of thumb is: number of grid points * $1.2 \cdot 10^{-4}$ seconds; this number refers to a AMDAHL 470 V/6 -II computer. The following Table lists CPU- estimates for a couple of grid sizes n (square grid). The estimates refer to a AMDAHL 470 V/6-II computer:

n	CPU-time (sec)
5	0.005
10	0.012
25	0.074
50	0.288
100	1.162

Table 4.1.2

4.2 Smooth surface interpolation/differentiation

How a bicubic spline is interpolated and differentiated, is demonstrated in subsection 2.6.3; the formulas (2.6-7a, b) are optimal in terms of computer time (PART A).

The interpolation/differentiation of a spline is a 3-stage process:

In the first step the grid element, corresponding to the coordinates of the calculation point, has to be found: with (x_1, y_1) the coordinates of the lower left grid point and (h_x, h_y) the grid distances in x- and y- direction, the grid element indices are, for a point (x, y), simply given by

$$m = \text{int}((x-x_1)/h_x) + 1, \quad n = \text{int}((y-y_1/h_y) + 1,$$

(int...integer), if the calculation point (x, y) is located within the domain of definition of the bicubic spline.

The second step consists in the calculation of the 16 coefficients of the bicubic element. A straightforward way would be a calculation using equations (2.6-5a, b); which gives the 4x4 matrix of coefficients as a product of three 4x4 matrices (see PART A).

$$A = H^T(h_x)FH(h_y) .$$

This operation involves 128 multiplications and 128 additions and uses, even in the normalized version ($h_x = h_y = 1$), 700 micro-seconds to calculate all 16 coefficients (the matrix A). Since A has to be calculated, in general, for each calculation point, it is very important to optimize this algorithm in terms of CPU-time. After many trials I found a very fast and probably optimal solution which involves no multiplication and only 58 additions; the time elapsed for the calculation of all 16 coefficients using this fast algorithm is as little as 58 micro-seconds which corresponds to a 12-fold gain in calculation speed.

The following Table lists CPU-estimates for the calculation of all 16 coefficients for all elements of varying size square grids (n x n elements). The estimates refer to a AMDAHL 470 V/6-II computer:

n	CPU-time (sec)
5	0.0015
10	0.0058
25	0.0363
50	0.1450
100	0.5800

TABLE 4.2.1

The subroutine BILDE is responsible for the calculation of the coefficients.

After these two steps the actual interpolation/differentiation can be performed. Since all calculations in the bicubic spline algorithms refer to a normalized grid of unit grid distances, the relative coordinates of the calculation point are also to be referred to this unit grid distance; for a point (x, y) they are simply given by

$$\bar{x} = (x-x_1)/h_x - (m-1), \quad \bar{y} = (y-y_1)/h_y - (n-1)$$

with (m, n) element indices as defined above. The interpolated/differentiated bicubic spline at a point (x, y) (within the domain of definition) can then be obtained by the set of formulas (2.6-7a, b) of PART A after division by the appropriate grid distances. The calculation itself is performed in the subroutine BSFC. The subroutine is designed such that it can provide all spline derivatives

$$\frac{\partial^{\alpha} f(x,y)}{\partial x^{\alpha_1} \partial y^{\alpha_2}}, \qquad \alpha = \alpha_1 + \alpha_2 \leq 4, \qquad \alpha_1, \alpha_2 \leq 2$$

(BSFC is not designed for third order derivatives with respect to x and/or y.)

The following Table gives a listing of CPU-time estimates for the interpolation/differentiation part and for the total CPU-time used (index finding, calculation of parameters, interpolation/differentiation).

The following Figure 4.2.1 with the corresponding Table 4.2.3 shows the function values of 25 regularly distributed data, the corresponding spline surface, and lists interpolated/differentiated values. The program below has been used to generate the output given in Table 4.2.3.

```
C     A BICUBIC SPLINE INTERPOLATION/DIFFERENTIATION EXAMPLE
C     A(...,.) ... ARRAY STORING THE BICUBIC SPLINE DEFINING VALUES
C     MXMAX, MYMAX ... MAXIMUM 1. AND 2. DIMENSION OF A
C     MX, MY ... ACTUAL 1. AND 2. DIMENSION OF A
C     DX, DY ... GRID DISTANCES IN X AND Y - DIRECTION (ACROSS, ALONG)
C     XL, XU ... LOWER AND UPPER X - BOUNDS OF THE DOMAIN
C     YL, YU ... LOWER AND UPPER Y - BOUNDS OF THE DOMAIN
C     NC ... NUMBER OF CALCULATION POINTS PROCESSED
C     X, Y .. COORDINATES OF THE CALCULATION POINT (ACROSS, ALONG)
C     XNR, YNR ... NORMALIZED RELATIVE COORDINATES FOR BSFC
C     IX, IY ... NUMBER OF DERIVATIVES IN X AND Y - DIRECTION
      DIMENSION A(5,5,4)
      COMMON /COM2/XNR,YNR,DX,DY
      MXMAX=5
      MYMAX=5
      MX=5
      MY=5
      DX=2.5
      DY=4.
C     (XL,XU; YL,YU) ARE THE DOMAIN LIMITS
      XL=1.
      YL=-1.
      XU=XL+(MX-1)*DX
      YU=YL+(MY-1)*DY
C     PRINT DOMAIN AND GRID INFORMATION
      WRITE(6,6003) XL,XU,YL,YU,DX,DY,MX,MY
C     READ THE BICUBIC SPLINE DEFINING VALUES; HERE THEY ARE GENERATED
      DO 1 I=1,MX
      DO 1 J=1,MY
1     A(I,J,1)=(SIN(I*1.)+COS(J*2.))*10
      CALL BISP(A,MXMAX,MYMAX,MX,MY)
C     READ THE NUMBER OF CALCULATION POINTS
      READ(5,*) NC
      WRITE(6,6002)
      M=0
      N=0
      DO 2 I=1,NC
C     READ ALL CALCULATION POINT COORDINATES AND THE NUMBER OF DIFFEREN-
C     TIATIONS IN X(ACROSS) AND Y(ALONG) DIRECTION
      READ(5,*) X,Y,IX,IY
C     IF A CALCULATION POINT HAPPENS TO BE OUTSIDE THE DOMAIN, PRINT A
C     MESSAGE AND GO TO THE NEXT POINT
      IF(X.GE.XL.AND.X.LT.XU.AND.Y.GE.YL.AND.Y.LT.YU) GOTO 3
      WRITE(6,6000) I,X,Y,IX,IY
      GOTO 2
3     CONTINUE
C     CALCULATE THE ELEMENT INDICES AND THE NORMALIZED COORDINATES
      XN=(X-XL)/DX
      MN=INT(XN)
      XNR=XN-MN
      MN=MN+1
      YN=(Y-YL)/DY
      NN=INT(YN)
      YNR=YN-NN
      NN=NN+1
C     CHECK IF THE INDICES HAVE CHANGED RELATIVE TO THE LAST CALCULATION
C     POINT; IF NOT, THE POLYNOMIAL COEFFICIENTS NEED NOT BE CALCULATED
C     AGAIN
      IF(M.EQ.MN.AND.N.EQ.NN) GOTO 4
      M=MN
      N=NN
C     CALCULATE THE COEFFICIENTS OF THE BICUBIC POLYNOMIAL
      CALL BILDE(A,MXMAX,MYMAX,M,N)
4     CONTINUE
C     INTERPOLATION/DIFFERENTIATION
      F=BSFC(IX,IY)
C     PRINT THE RESULTS
      WRITE(6,6001) I,X,Y,IX,IY,F
2     CONTINUE
6000  FORMAT(1H0,4X,I5,2X,2F10.2,5X,2I3,2X,'   OUTSIDE DOMAIN')
6001  FORMAT(1H0,4X,I5,2X,2F10.2,5X,2I3,2X,F10.2)
6002  FORMAT(1H0,4X,'     I          X          Y          IX IY        F',//
     *)
6003  FORMAT(1H1,4X,'     DOMAIN LIMITS AND GRID PARAMETERS',//,9X,'XL ..
     *.',F10.2,5X,'XU ...',F10.2,/,9X,'YL ...',F10.2,5X,'YU ...', F10.2,
     */,9X,'DX ...',F10.2,5X,'DY ...',F10.2,/,9X,'MX ...',I10,5X,'NY ...
     *',I10,///)
      STOP
      END
```

Program corresponding to Table 4.2.3

α_1	α_2	only inter-pol/diff.	total CPU-time (microsec.)
0	0	18	76
1	0	19	77
0	1	19	77
2	0	18	76
1	1	21	79
0	2	28	76
2	1	20	78
1	2	20	78
2	2	19	77

Table 4.2.2

CPU - time estimates for 2-D spline interpolation/differentiation

DOMAIN LIMITS AND GRID PARAMETERS

XL ...	1.00	XU ...	11.00
YL ...	-1.00	YU ...	15.00
DX ...	2.50	DY ...	4.00
MX ...	5	NY ...	5

I	X	Y	IX	IY	F
1	3.27	7.38	0	0	19.12
2	8.53	2.11	1	0	-2.20
3	1.16	10.24	0	1	-4.04
4	13.15	7.55	2	0	OUTSIDE DOMAIN
5	1.10	14.12	1	1	0.00
6	9.99	2.53	0	2	2.31
7	6.34	12.21	2	1	0.00
8	7.51	8.77	1	2	-0.00
9	-7.22	10.00	2	2	OUTSIDE DOMAIN

Table 4.2.3: Bicubic spline interpolation/differentiation

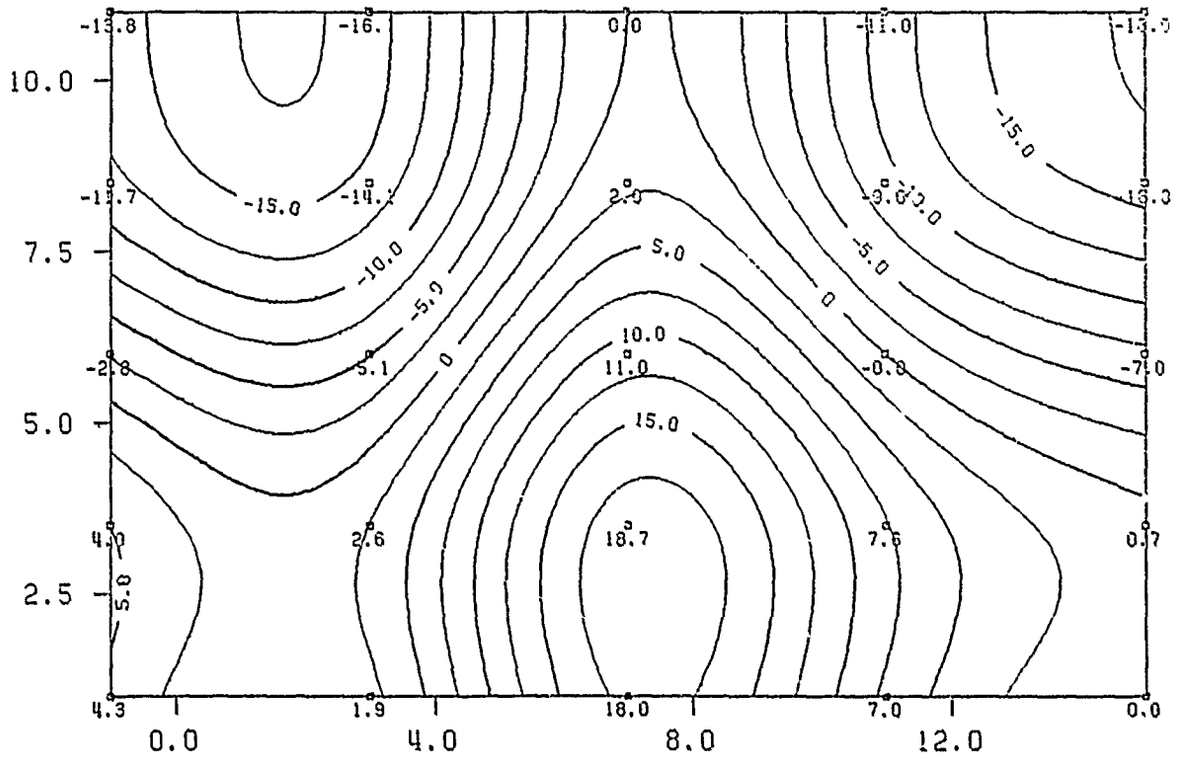


Figure 4.2.1: Data distribution and corresponding bicubic spline surface contour plot.

4.3 Data approximation errors

Whenever the function values of the surface are predicted at the grid points of a regular rectangular grid, and the surface is represented by a bicubic spline, interpolating these grid values, the irregularly distributed homogenous data are, in general, not reproduced. It is obvious that the data reproduction errors (the difference between the actual data f_i and the corresponding surface value $\hat{f}_i$) decrease with decreasing grid distance; if the error norm is defined in terms of the maximum error,

$$||e|| := \max_{\forall_i} |f_i - \hat{f}_i|,$$

the error norm goes to zero. This, however, does not necessarily mean that the spline representation of the whole surface is getting better with decreasing grid distance. Nevertheless, the above error norm provides a first estimate of how well the data are reproduced.

The subroutine REPRO is designed for the calculation of the individual data reproduction errors, the average absolute, and the root mean square (RMS) approximation error. Whenever the individual errors are calculated, the original function values will be destroyed and replaced by the corresponding approximation (reproduction) errors. If a data point happens to be located outside the surface domain, it will not be considered for the data reproduction error calculation and a value of 99999.99 will be assigned to its error. The average and RMS errors are edited on the lineprinter (or any other assigned output device). The following example may illustrate an isolated use of the subroutine REPRO. Further information can be found in the comment statements to REPRO.

```
C     TEST OF THE DATA REPRODUCTION ERROR CALCULATION PROGRAM 'REPRO'
      DIMENSION A(25,25,4), ZZ(100)
      COMMON /XDAT/X(100) /YDAT/Y(100) /ZDAT/Z(100) /UCIP/IUP(30)
     */UCRP/UP(30) /IOSPAR/NOUT
      DATA MXMAX,NYMAX,MZMAX/2*25,4/
C     INPUT UNIT ... INPUT, OUTPUT UNIT ... NOUT
      INPUT=5
      NOUT=6
C     READ THE DATA
C     READ THE DATA, ERROR INDICATES END OF DATA SET, COPY Z(.)
C     ONTO ZZ(.)
      N=1
18    READ(INPUT,*,ERR=19) X(N), Y(N), Z(N)
      ZZ(N)=Z(N)
      N=N+1
      GOTO 18
19    NDAT=N-1
      WRITE(NOUT,5001) NDAT
5001  FORMAT(1H0,4X,'NUMBER OF DATA FOUND IN DATA SET ... ',I5,/)
C     READ AND ECHO SURFACE ARRAY PARAMETERS
      READ(5,*) IUP(2), IUP(3), (UP(I),I=1,4)
      WRITE(NOUT,5002) (UP(I),I=1,4), IUP(2), IUP(3)
5002  FORMAT(1H0,4X,'SURFACE ARRAY PARAMETERS :',/,5X,
     *'COORDINATES OF LOWER LEFT SURFACE ARRAY POINT : X = ',F10.2,
     *'   Y = ',F10.2,/,5X,'GRID DISTANCES IN X-AND Y-DIRECTION :',
     *2F10.4,/,5X,'NUMBER OF GRID POINTS IN X-AND Y-DIRECTION :',
     *2I5,//)
C     READ THE SURFACE ARRAY A(.,.,1) AND ECHO IT
      READ(5,*) ((A(I,J,1),J=1,IUP(3)),I=IUP(2),1,-1)
      WRITE(NOUT,6000)
6000  FORMAT(1H0,4X,'FUNCTION VALUES AT THE GRID POINTS :',//)
      CALL DRUCK(A,MXMAX,NYMAX,MZMAX,IUP(2),IUP(3),1,UP(1),UP(2),UP(3),
     *UP(4),36,8,0,0,0)
C     CALCULATE THE BICUBIC SPLINE REPRESENTATION
      CALL BISP(A,MXMAX,NYMAX,IUP(2),IUP(3))
C     CALCULATE THE DATA REPRODUCTION ERRORS
      CALL REPRO(A,MXMAX,NYMAX,NDAT,3)
C     PRINT THE DATA AND THE INDIVIDUAL ERRORS
      WRITE(NOUT,7000)
7000  FORMAT(1H0,4X,'SPLINE DATA REPRODUCTION ERRORS :',/,
     *5X,'    I       X(I)       Y(I)       Z(I)     ERR(I)',/)
      DO 7001 II=1,NDAT
7001  WRITE(NOUT,7002) II, X(II), Y(II), ZZ(II), Z(II)
7002  FORMAT(1H ,4X,I5,4F10.2)
      STOP
      END
```

NUMBER OF DATA FOUND IN DATA SET ... 17

SURFACE ARRAY PARAMETERS :
COORDINATES OF LOWER LEFT SURFACE ARRAY POINT : X = 47.67 Y = 14.00
GRID DISTANCES IN X-AND Y-DIRECTION : .0417 .0667
NUMBER OF GRID POINTS IN X-AND Y-DIRECTION : 8 8

FUNCTION VALUES AT THE GRID POINTS :

lat. \ long.	14 .0	14 4.0	14 8.0	14 12.0	14 16.0	14 20.0	14 24.0	14 28.0
47 57.5	-11.39	-12.17	-9.12	-12.18	-11.53	-8.71	-10.20	-10.15
47 55.0	-10.25	-9.21	-9.04	-11.25	-11.53	-9.78	-10.73	-10.96
47 52.5	-13.25	-13.83	-14.62	-14.63	-11.81	-12.33	-12.01	-13.79
47 50.0	-13.26	-13.23	-11.79	-11.55	-15.02	-10.69	-12.37	-12.05
47 47.5	-13.51	-15.99	-18.65	-28.03	-20.28	-20.53	-20.01	-18.73
47 45.0	-18.50	-19.75	-20.72	-19.62	-28.49	-18.33	-15.81	-14.29
47 42.5	-17.15	-17.06	-19.28	-24.45	-19.44	-19.45	-19.78	-19.54
47 40.0	-18.62	-18.79	-19.31	-24.48	-22.72	-18.64	-16.89	-17.44

AVERAGE APPROXIMATION ERROR ... .723
RMS APPROXIMATION ERROR ... .848

SPLINE DATA REPRODUCTION ERRORS :

I	X(I)	Y(I)	Z(I)	ERR(I)
1	47.88	14.26	-11.35	.46
2	47.84	14.26	-13.46	.93
3	47.77	14.17	-22.93	-.03
4	47.72	14.33	-20.64	-1.63
5	47.93	14.25	-12.36	-.53
6	47.96	14.01	-18.02	100000.00
7	47.70	14.16	-21.37	.93
8	48.15	14.43	-11.33	100000.00
9	48.19	14.36	-14.02	100000.00
10	47.75	14.34	-17.04	.46
11	47.79	14.23	-24.51	1.26
12	47.93	14.07	-9.38	-.20
13	47.92	14.52	-10.70	100000.00
14	48.16	14.53	-10.84	100000.00
15	47.69	14.19	-24.44	.88
16	47.77	14.11	-19.10	.64
17	48.18	14.03	-17.19	100000.00

5. Axis plot

Axes are plotted automatically by GSPP in connection with profile and contour plots, unless its plotting is suppressed. The subroutine ACHSE is responsible for the axis plot. It is designed for virtually all different cases: arbitrary direction, scaling, tick marks right-or leftbound, scale numbers right or left of axis in four different directions (integer multiples of 90°), variable number of decimal digits, variable height of scale numbers, variable tick mark length, variable distance (axis, scale numbers), and many other options more. For a detailed reference see the comment of the program listing.

In the following three examples of axis plots are shown. Axis n corresponds to the n'th call of ACHSE in the subsequently listed test program.

```
C      EXAMPLES OF DIFFERENT AXIS PLOTS
C      A DESCRIPTION OF THE INPUT (AND OUTPUT) PARAMETERS CAN BE FOUND
C      IN THE SUBROUTINE 'ACHSE'
       CALL PLOTS(0.,0.,10)
C      PLOT UNIT :
       CALL ACHSE(0.,9.,-15.,20.,5.,2,0.,5.,0.2,0.5,1.,1.,0.3,1.,-1,0,1,
     *1,DX0,NTM)
       CALL ACHSE(0.,11.,100.,200.,25.,2,60.,20.,0.25,-0.4,1.,1.,0.25,1.
     *  1,1,1,0,DX0,NTM)
       CALL ACHSE(0.,15.,1500.,3500.,500.,2,300.,250.,-0.3,0.5,1.,1.,0.3,
     *1,-1,1,1,1,DX0,NTM)
       CALL PLOT(0.,0.,999)
       STOP
       END
```

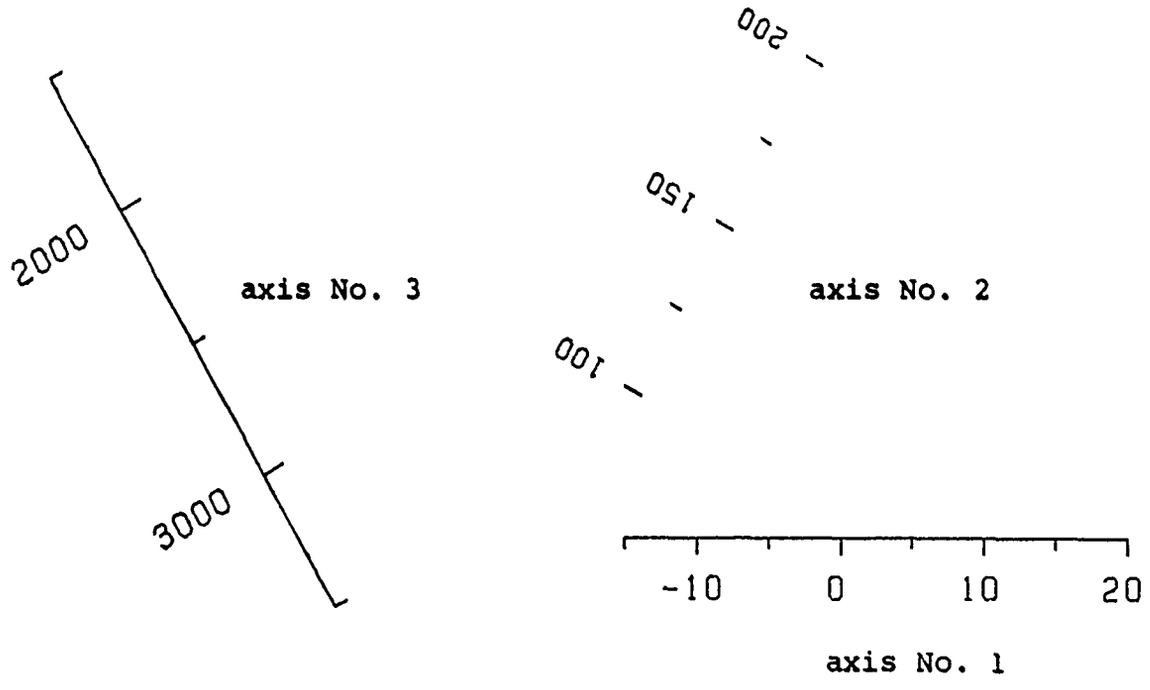


Figure 5.1: Examples of axis plots

6. Grid plot

Superimposing a grid over a contour plot simplifies a possible interpolation process. GSPP assures that the individual grid interval is identical with the tick mark interval, if an integrated grid plot is requested. But a grid of arbitrary shape, size, and orientation can be superimposed also externally. This is done by the subroutine GRIPLO. A number of options can be chosen by the user: A rectangular grid can be full line grid, a dashed line grid, or simply consist of open crosses at the grid points (intersections of grid lines); the grid can be arbitrarily oriented, can be shifted arbitrarily relative to the lower left corner of the grid area, the grid distances in x- and y- direction are arbitrary, the plot of a bounding rectangle (representing the grid area) can be suppressed, and other options. For a detailed information see the comment statements in the sub-routine GRIPLO. Some grid examples are shown below.

```
C     EXAMPLES OF DIFFERENT GRID PLOTS
C     A DESCRIPTION OF INPUT PARAMETERS CAN BE FOUND IN THE SUBROUTINE
C     'GRIPLO'
      CALL PLOTS(0.,0.,10)
C     PLOT UNIT : CENTIMETER
      CALL GRIPLO(1.,9.,8.,6.,0.8,0.6,2.,1.5,4,4,0.5,0.,0.,0,1)
      CALL GRIPLO(11.,10.,8.,6.,0.,0.,1.,2.,9,4,0.2,1.,-20.,0,1)
      CALL GRIPLO(1.,1.,8.,6.,1.,1.,1.,1.,9,7,0.1,1.5,0.,1,0)
      CALL GRIPLO(11.,1.,8.,6.,0.5,0.,2,2.5,4,3,-0.2,2.,0.,0,1)
      CALL PLOT(0.,0.,999)
      STOP
      END
```

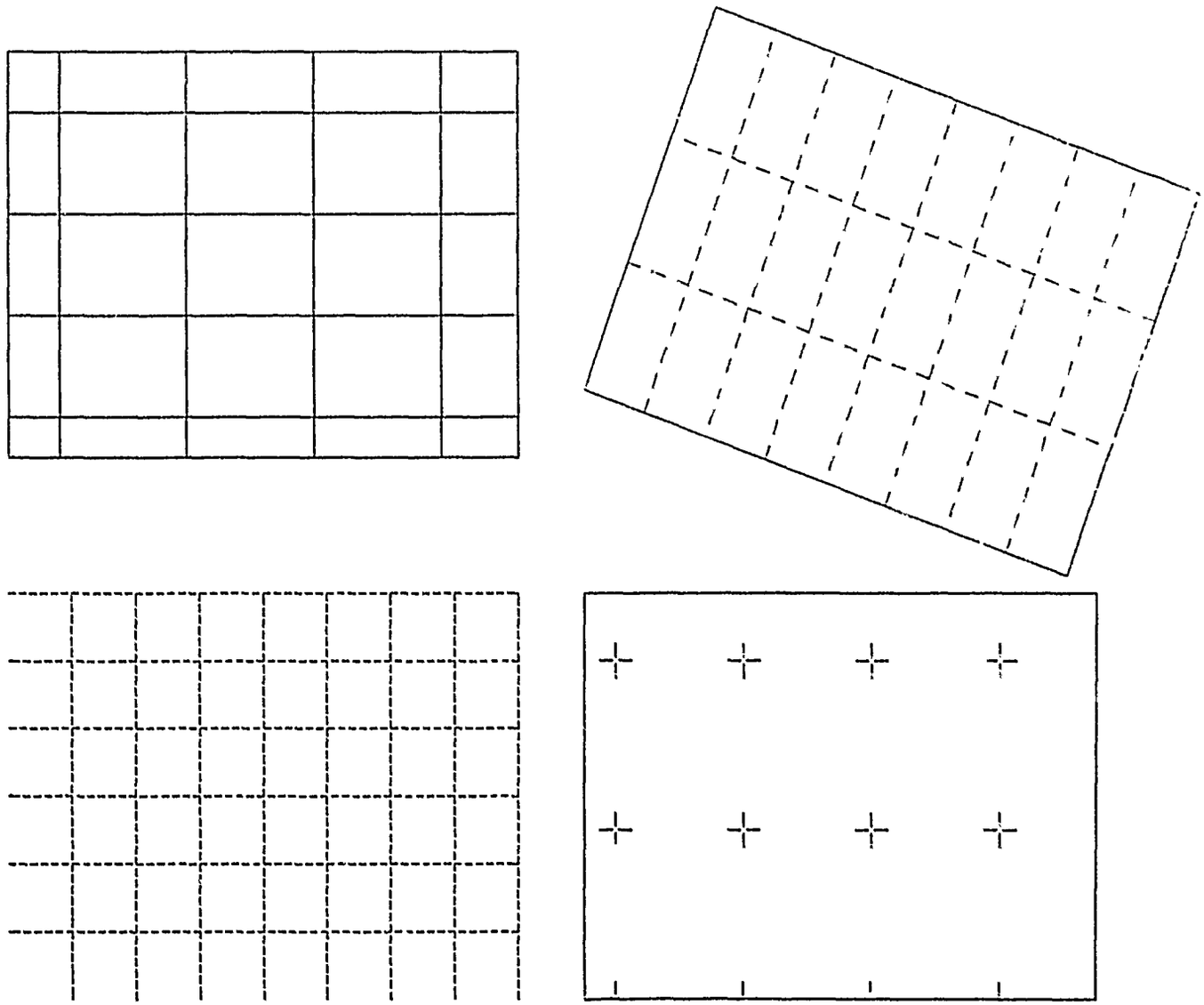


Figure 6.1: Examples of grid plots

7. Title plot

The description of plots (bead, axis labels) is performed by the subroutine TITLE, a part of GSPP. The main purposes of this program are:

a) find the number of lines of alphanumeric characters which belong to the title (in connection with GSPP, a maximum of 10 lines is allowed; in isolated applications, a maximum of 100 lines can be plotted by a single call);

b) find the number of alphanumeric characters per line (max. =80) and the maximum number of characters;

c) find the maximum actual title length and height, compare it with the corresponding allowed maxima, and reduce the symbol height, if necessary.(*)

d) Shift the title lines; four different title line patterns can be obtained: the title lines can be plotted as appearing on the punched cards (or any other input device); the title lines can be shifted to the left (leftbound); the title lines can be shifted to the right (right-bound); the title lines can be centered.

Moreover, the title can be put into a rectangular frame, called title boundary rectangle; the title can be plotted in any direction (0^{o}-360^{o}), 0^{o} is the horizontal mode; when plotting on an electrostatic plotter like the Versatec, the line width can vary between single and 5-fold linewidth. The title itself is stored on a 2-dimensional array, each column representing a 80-character string (1 card).

Some typical examples are shown below. For more information (title and parameter transfer) see the comments in the subroutine TITLE.

(*) If a symbol height reduction is necessary, a message will be edited on the assigned output unit.

```
C       EXAMPLES OF DIFFERENT TITLE PLOTS
        DIMENSION TIT(20,10)
        COMMON /UTITPA/Y,SHNR,THMAX,XLTH,ROT,IPEN,IC,IW3
       *       /TITAUX/WH,CMINCH,TH,TITLTH,X,ICV,LC,NRL,MAX /IOSPAR/IO
        IO=6
C       READ THE NUMBER OF  TITLE LINES
        READ(5,*) NTL
C       READ THE TITLE PARAMETERS
        READ(5,*) X,Y,SHNR,THMAX,WH,XLTH,ROT,CMINCH,ICV,IW3,IPEN,IC,LC
C       READ THE TITLE  (EACH COLUMN OF TIT(.,.) CORRESPONDS TO ONE (1)
C       TITLE LINE
        READ(5,5) ((TIT(I,J),I=1,20),J=1,NTL)
5       FORMAT(20A4)
        CALL PLOTS(0.,0.,10)
        CALL TITLE(TIT,10)
        IW3=0
        IPEN=3
        ROT=90.
        Y=9.
        CALL TITLE(TIT,10)
        IC=1
        IPEN=2
        ROT=0.
        Y=5.
        CALL TITLE(TIT,10)
        IC=0
        IPEN=1
        Y=0.5
        CALL TITLE(TIT,10)
        CALL PLOT(0.,0.,999)
        STOP
        END
```

input parameters:

5

0., 20., 0.3, 5., 1., 18., 0., 0.3937, 0, 1, 5, 11, 0

the 5 title lines

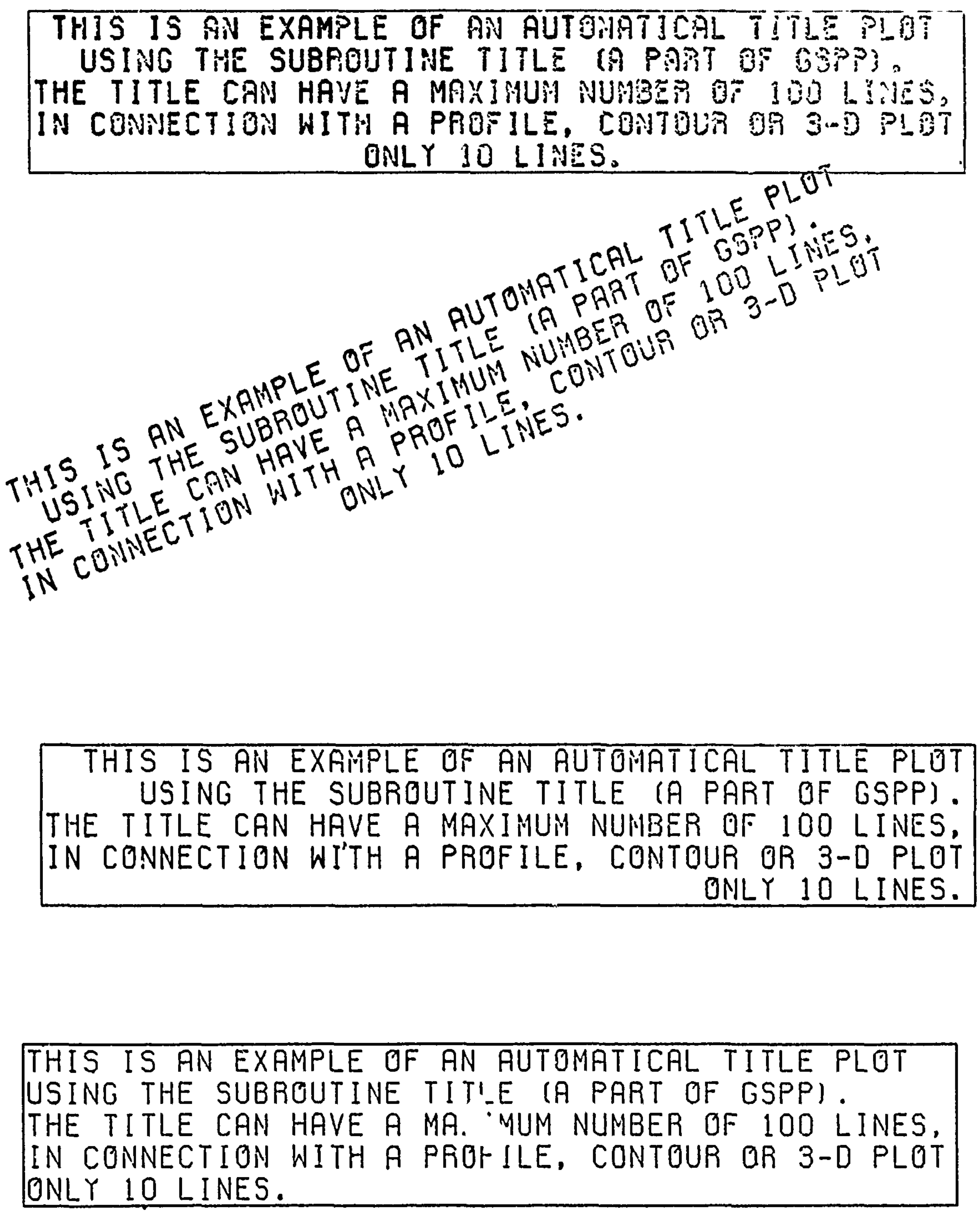


Figure 7.1: Examples of title plots
(The title and the top correspond to the first call, the title at the bottom to the last call of the subroutine TITLE.)

8. Range division

A particularly useful element out of GSPP, is the subroutine RNGDIV; it is frequently used in GSPP for the purpose of replacing the human decision process like finding "optimal" contour intervals, "optimal" grid distances, or "optimal" tick mark distances. "Optimal" is interpreted in terms of a most reasonable unbiased decision.

The main objectives of RNGDIV are as follows: given a range on the real number line by a start- and endpoint (lower and upper bound); the range should be divided into intervals of the form

$$(0.1,\ 0.25,\ 0.5) * 10^{k}$$

with k an integer; the maximum number of intervals can be chosen by the user; there is the possibility of adjusting the interval start- and endpoint such that the interval start- and endpoints have "coordinates" of the form

$$(\alpha * 0.1,\ \beta * 0.25,\ \gamma * 0.5) * 10^{k}$$

with $\alpha = 1, \ldots,$; $\beta = 1, 2, 3$; $\gamma = 1, 2$. The adjustment can be a range extension or a range contraction (e.g. in contouring). The program returns, apart from the calculated interval length and the number of intervals, also the recommended number of decimal places for a graphical representation of the scale numbers. (Recommended is: 3 significant digits, if the maximum scale number is greater than 100, only integer representation of the real scale number.) A special application is the estimation of the number of significant digits for a specified range.

In the sequal a couple of range division examples, calculated using RNGDIV, are listed. Further information can be found in the comment statements to RNGDIV.

```
C       TEST OF THE SUBROUTINE 'RNGDIV'. FOR DETAILS SEE THE
C       COMMENTS TO 'RNGDIV'
        INPUT=5
        NOUT=6
        WRITE(NOUT,5)
1       READ(INPUT,*,ERR=2) XA,XE,NRIM,IN
        CALL RNGDIV(XA,XE,XLA,XLE,DXL,NRIM,IN,NRI,ND)
C       PRINT INPUT AND OUTPUT
        WRITE(NOUT,4) XA,XE,XLA,XLE,DXL,NRIM,IN,NRI,ND
        GOTO 1
5       FORMAT(1H ,4X,'TEST OF SUBROUTINE RNGDIV',//,5X,
     *'      XA          XE          XLA         XLE       ',
     *'  DXL    NRIM    IN   NRI   ND',//)
4       FORMAT(1H ,4F12.2,4I5)
2       STOP
        END
```

```
.TRNGDIV

 OF SUBROUTINE RNGDIV
   XA          XE          XLA         XLE         DXL    NRIM    IN   NRI   ND

-112.50      273.40      -125.00      250.00      25.00     20    -1    14    -1
-112.50      273.40      -125.00      275.00      25.00     20     1    16    -1
-112.50      273.40      -125.00      275.00      25.00      0     0    16    -1
```

9. Clipped boundary line plot

In connection with contour plots in regions of arbitrary shape it is often requested to plot the boundaries of these regions with the restriction that the boundary lines should be clipped off at its intersections with the boundary of the rectangular plotting domain.

The subroutine BNDPLO is designed for this purpose. It accepts, in principle, an arbitrary number of (district) boundary lines which can be plotted with different line widths on an electrostatic plotter or in different colors on a multi-color plotter.

In order to clip off the line plot at the boundary of the rectangular domain, each line element passes a procedure like that described in section 2.8.5 which provides sufficient information about

the location of the line element relative to the rectangular domain. The actual intersection uses a simple line intersection algorithm. If so desired, the boundaries can be mapped according to an arbitrary mapping equation, $\bar{x} = \bar{x}(x, y)$, $\bar{y} = \bar{y}(x, y)$, which has to be provided by the user (subroutine GSPROJ).

The following example may illustrate the use of the subprogram.

```
C     TEST OF THE SUBROUTINE 'BNDPLO'. FOR DETAILS SEE THE
C     COMMENTS TO 'BNDPLO'.
      DIMENSION FORMD(10)
      COMMON /XBOUND/X(100) /YBOUND/Y(100) /BOUINF/NAE(20)
     *       /BOUPEN/NPEN(10) /CCLIP/XL,XU,YL,YU
      INPUT=5
      NOUT=6
C     READ AND ECHO THE NUMBER OF BOUNDARIES, ITS START- AND
C     ENDPOINT LOCATIONS ON THE VECTORS X(.) AND Y(.) AND THE
C     CORRESPONDING PENWIDTHS NPEN(.)
      READ(INPUT,*) NAE(1)
      N=NAE(1)
      NB=2*NAE(1)+1
      READ(INPUT,*) (NAE(I),I=2,NB)
      READ(INPUT,*) (NPEN(I),I=1,N)
      WRITE(NOUT,10)
10    FORMAT(1H0,4X,'BOUNDARY LINE INFORMATION VECTOR NAE(.)',
     */,5X,'    I    NAE(I)  NAE(I+1)  NPEN(I/2)',//)
      DO 1 I=2,NB,2
      I1=I+1
      I2=I/2
1     WRITE(NOUT,11) I,NAE(I),NAE(I1),NPEN(I2)
      N=NAE(I1)
11    FORMAT(1H ,4X,I5,I9,I8,I10)
C     READ INPUT FORMAT FOR BOUNDARY COORDINATES
      READ(INPUT,16) FORMD
16    FORMAT(10A4)
C     READ AND ECHO THE BOUNDARY COORDINATES
      WRITE(NOUT,12)
12    FORMAT(1H0,4X,'BND#     I          X(I)      Y(I)',//)
      N=NAE(1)
      DO 2 I=1,N
      IS=NAE(2*I)
      IE=NAE(2*I+1)
      DO 2 J=IS,IE
      READ(INPUT,FORMD) X(J),Y(J)
2     WRITE(NOUT,13) I,J,X(J),Y(J)
13    FORMAT(1H ,4X,2I5,5X,2F10.2)
C     READ AND ECHO LOWER AND UPPER X- AND Y-COORDINATES OF
C     THE RECTANGULAR PLOTTING DOMAIN
      READ(INPUT,*) XL,XU,YL,YU
      WRITE(NOUT,14) XL,XU,YL,YU
14    FORMAT(1H0,4X,'PLOTTING DOMAIN LIMITS :',//,5X,
     *'XLOWER ... ',F10.2,5X,'XUPPER ... ',F10.2,/,5X,
     *'YLOWER ... ',F10.2,5X,'YUPPER ... ',F10.2)
C     READ AND ECHO PLOT PARAMETERS (ORIGIN, SCALE)
      READ(INPUT,*) X0,Y0,FAC
      WRITE(NOUT,15) X0,Y0,FAC
15    FORMAT(1H0,4X,'PLOT PARAMETERS :',//,5X,'X0 ...  ,
     *F10.2,5X,'Y0 ... ',F10.2,5X,'FAC ... ',F10.2)
C     INITIALIZE PLOT
      CALL PLOTS(0.,0.,10)
      CALL FRAME(0,0.,100.,0.,70.)
C     PLOT THE BOUNDARIES
      CALL BNDPLO(X0,Y0,FAC,0,0,0)
C     PLOT THE BOUNDARY RECTANGLE OF THE PLOTTING DOMAIN
      CALL RECT(Y0,X0,(YU-YL)/FAC,(XU-XL)/FAC,0.,1)
C     STOP PLOT
      CALL PLOT(0.,0.,999)
      STOP
      END
```

Input data to the above program

```
1       5
2       1,9,10,26,27,35,36,40,41,47
3       1,2,3,1,2
4       (9X,2F10.2)
5       1    2     45.80      -21.80
6       1    3     41.60      -16.40
7       1    4     41.80      -9.40
8       1    5     45.20      -15.20
9       1    6     50.60      -14.60
10      1    7     54.00      -17.20
11      1    8     59.20      -18.00
12      1    9     50.00      -18.80
13      1    10    45.80      -21.80
14      2    14    55.00      5.20
15      2    15    62.00      1.80
16      2    16    68.00      -4.60
17      2    17    84.60      -6.00
18      2    18    86.40      -3.80
19      2    19    88.20      -0.80
20      2    20    88.20      1.40
21      2    21    81.80      6.40
22      2    22    78.00      11.60
23      2    23    78.00      21.20
24      2    24    75.60      24.00
25      2    25    62.40      22.60
26      2    26    50.40      25.20
27      2    27    45.80      23.60
28      2    28    46.00      12.80
29      2    29    49.80      7.80
30      2    30    55.00      5.20
31      3    36    71.40      11.80
32      3    37    75.20      4.80
33      3    38    62.00      6.40
34      3    39    55.20      9.80
35      3    40    54.00      12.40
36      3    41    54.00      15.40
37      3    42    64.80      16.20
38      3    43    69.20      19.20
39      3    44    71.40      11.80
40      4    45    39.40      35.60
41      4    46    54.80      32.00
42      4    47    55.00      40.20
43      4    48    50.60      35.40
44      4    49    39.40      35.60
45      5    52    103.40     22.60
46      5    53    84.80      33.40
47      5    54    68.00      28.00
48      5    55    61.80      29.80
49      5    56    63.00      35.00
50      5    57    79.80      39.80
51      5    58    103.40     22.60
52      25.2,92.2,-12.5,40.0
53      3.,3.,5.,
```

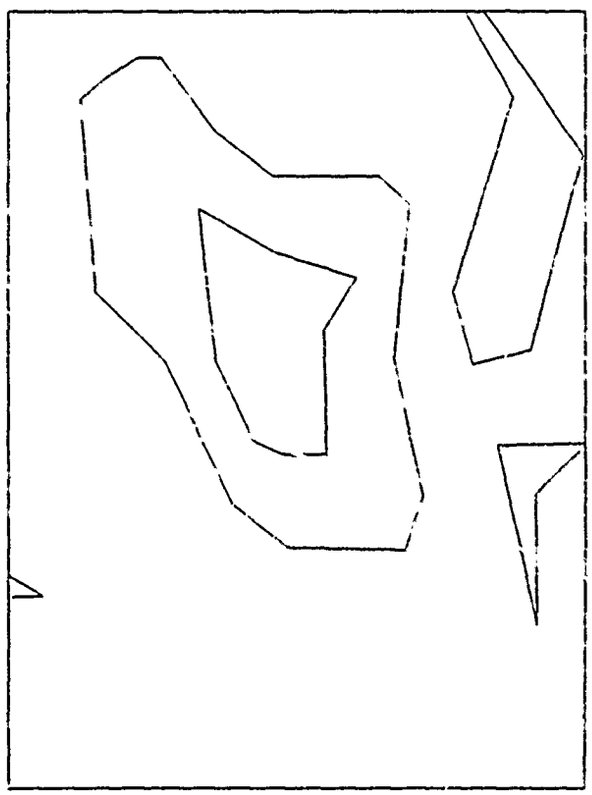


Figure 9.1 Example of clipped boundary line plot corresponding to the above input data

REFERENCES

Adigüzel, M. (1979): Digital terrain models and their applications to perspective diagramming and contouring. Thesis at the Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Ahlberg, J.H., E.N. Nilson, and J.L. Walsh (1967): The Theory of Splines and Their Applications. Academic Press, London-New York.

Bhattacharyya, B.K. (1969): Bicubic spline interpolation as a method for treatment of potential field data. Geophysics, Vol. 34, No. 3.

Bjerhammar, A. (1973): Theory of Errors and Generalized Matrix Inverses. Elsevier Publishing Company, Amsterdam.

Brigham, E.O. (1974): The Fast Fourier Transform: Englewood Cliffs, New Jersey.

Bybee, J.E. and G.M. Bedross (1978): The IPIN computer network control softward. In: Proceedings of the "Digital Terrain Models Symposium", St. Louis, Missouri.

Davis, P.J. (1975): Interpolation and Approximation. Dover Publications, Inc., New York.

Faddejew, D.K. and W.N. Faddejewa (1970): Numerische Methoden der Linearen Algebra. Oldenbourg-Verlag, München.

Gradshteyn, I.S. and I.W. Ryzhik (1971): Table of Integrals, Series, and Products. Academic Press, New York.

Gray, R.M. (1971): Toeplitz and circulant matrices: A review. Technical Report No. 6502-1, Stanford Electronics Laboratories, Stanford, California.

Heiskanen, W.A. and H. Moritz (1967): Physical Geodesy. W.H. Freeman and Company, San Francisco.

Lachapelle, G. (1977): Estimation of disturbing potential components using a combined integral formulae and collocation approach. Paper presented at the 2nd International Summer School in the Mountains, Ramsau, Austria, published in: Collected Papers Geodetic Survey, Surveys and Mapping Branch, Dept. of Energy, Mines and Resources Canada.

Moritz, H. (1978): The operational approach to physical geodesy. Report No. 277, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Moritz, H. (1980): Advanced Physical Geodesy. H. Wichmann Verlag, Karlsruhe.

Neubauer, H.G. (1978): Beiträge fur digitalen photogrammetrischen hessung für die Herotellung topographischer Karteu. Nachr. Kart. Verm, Reihe 1, Nr. 77.

Rapp, R.H. (1978): Results of the application of least-squares collocation to selected geodetic problems. In: Moritz, H. and H. Sünkel (edt.) (1978): Approximation Methods in Geodesy. H. Wichmann Verlag, Karsruhe.

Rapp, R.H. (1979): Global anomaly and undulation recovery using GEOS-3 altimeter data. Report No. 285, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Schoenberg, I.J. (1973): Cardinal spline interpolation. Series 12, Regional Conference Series in Applied Mathematics, published by Siam, Philadelphia, Pennsylvania.

Schumaker, L.L. (1976): Fitting surfaces to scattered data. In: G.G. Lorenz, C.K. Chui, L.L. Schumaker: Approximation Theory II. Academic Press, Inc., New York.

Schwarz, K.P. (1976): Geodetic Accuracies obtainable from measurements of first and second order gravitational gradients. Report No. 242, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Schwarz, K.P. and G. Lachapelle (1979): Local characteristics of the gravity anomaly covariance function. Bulletin Geodesique, in print.

Shepard, D. (1964): A two-dimensional interpolation function for irregularly spaced data. Proceedings of the 1964 A National Conference.

Späth, H. (1973): Spline-Algorithmen fur Konstruktion Glatter Kurven und Flächen. Oldenbourg-Verlag, München.

Sünkel, H. (1977a): A FORTRAN IV program to calculate and plot isolines. In: J. Krynski, H. Noë, K.P. Schwarz, H. Sünkel, Numerical studies and programs for interpolation and collocation. Folge 26, Mitteilungen der geodätischen Institute der TU Graz.

Sünkel, H. (1977b): Die Darstellung geodätischer Integralformeln durch bikubische Spline-Funktionen. Folge 28, Mitteilungen der geodätischen Institute der TU Graz.

Sünkel, H. (1978): Approximation of covariance functions by non-positive definite functions. Report No. 271, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Sünkel, H. (1979): A covariance approximation procedure. Report No. 286, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Sünkel, H. (1980): Cardinal interpolation. Report in print. Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Tscherning, C.C. (1974): A FORTRAN IV program for the determination of the anomalies potential using step-wise least-squares collocation. Report No. 212, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Tscherning, C.C. (1976): Covariance expressions for second and lower order derivatives of the anomalous potential. Report No. 225, Department of Geodetic Science, The Ohio State University, Columbus, Ohio.

Watkins, S.L. (1973): Masked three-dimensional plot program with rotations. Applied Research Laboratories, The University of Texas at Austin, Austin.